Parenthetical Constructions - an Argument against Modularity

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

Abstract

This paper presents an argument against modularizing linguistic information in natural language generation systems. We argue that complex linguistic constructions require grammatical information to be located in the same module, in order to avoid over-complicating the system architecture. We demonstrate this point by showing how parenthetical constructions — which have only been generated in previous systems using an aggregation or revision module — can be generated by a surface realizer when using an integrated grammar.

1 Introduction

The ultimate aim of research on natural language generation is to develop large-scale, domain independent NLG systems, which are able to generate high quality, fluent and well-formatted texts. Ideally the produced texts will be as long as needed to convey the information given in the input and should be presented in a style that is appropriate for the purposes of the user. Current NLG systems typically produce paragraph-length text tailored to a specific domain and the grammars in these systems contain only a limited number of grammatical constructions, typically collected during a corpus study of example documents. Often the grammar is implemented using schemas or "canned" expressions, and individual grammatical levels are distributed in independent modules.

Organizing the grammar this way severely limits the flexibility of NLG systems. It has long been recognized in the literature that text fluency can be improved by modeling interactions between grammar modules. The most commonly mentioned interactions are those among discourse/rhetorical relations and syntax (Scott and Souza, 1990; Hovy, 1993; Callaway, 2003), rhetorical relations, syntax and referring expressions (Kibble and Power, 2004); and layout and referring expressions (N. Bouayad-Agha, 2001). It is clear that in order to generate high quality, coherent discourse, a generator needs access to a grammar which is able to model the interdependent, context-sensitive behaviour of these separate linguistic phenomena.

In this paper we draw a parallel between grammar design and the design of natural language generation systems. We argue that in order to generate complex linguistic constructions, current NLG systems tend to have overly complicated architectures. To illustrate this point we show how a surface realizer can take on tasks from other components when linguistic information from different grammar modules (and hence, system modules) is integrated. This simplifies system architecture by reducing the need for interaction between modules and enables the generator to produce more complex and coherent text. We illustrate this point by first showing constraints that parenthetical constructions impose on pronominalization. Then we present a grammar which integrates a representation for referring expressions into a syntax/discourse grammar. Finally we show that using this grammar, we can generate complex, coherent paragraphs which contain parenthetical constructions using only a surface realizer.

2 The problem of generating parenthetical constructions

Parentheticals are constructions that provide less important or background information in texts and they are a prime example of interactions between referring expressions, syntax, layout and discourse structure. Parentheticals help readers distinguish between more and less important propositions and therefore significantly increase the fluency and readability of the generated text. Despite this ma-

Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks, ACL-IJCNLP 2009, pages 46–53,
Suntec, Singapore, 6 August 2009.

jor effect on the quality of the generated text, current natural language generation systems still do not have a principled way of producing parentheticals. In this paper we focus on parenthetical constructions which take the form of a subordinate clause introduced by a discourse connective. Some examples of this type of parentheticals in the Wall Street Journal are illustrated (1):

(1) a The irony is that the attack commercial, *after getting a boost in last year's presidential campaign,* has come of age in an off-off election year with only a few contests scattered across the country.
b the 1989 fall total of 80, *while well below 1988 activity,* shows a steady ratcheting up in citizen referenda and initiatives
c pollination, *while easy in corn because the carrier is wind,* is more complex and involves insects as carriers in crops such as cotton

The examples in (2) illustrate the difficulties in generating parenthetical constructions by showing some possible but *incoherent* realizations of the same message. In particular, they illustrate the importance of appropriate punctuation marks (2a), syntactic requirements of discourse connectives (2b), the limit on embedding (2c), and the importance of ordering syntactic arguments (thematic structure/information structure) (2d).

(2) a # The FDA though it bans Elixir since it contains Gestodene approves Elixir Plus.
b # The FDA – but it bans Elixir since it contains Gestodene – approves Elixir Plus.
c # The FDA though since Elixir contains Gestodene , it bans Elixir approves Elixir Plus.
d # The FDA, since Gestodene is an ingredient of Elixir, bans Elixir. But it approves Elixir Plus.

Correct realizations of the same message would include:

(3) a The FDA — though it bans Elixir since it contains Gestodene — approves Elixir Plus .
b The FDA bans Elixir because it contains Gestodene. However, Elixir Plus is approved by the FDA
c The FDA approves Elixir Plus although Elixir — since it contains Gestodene — is banned by the FDA.

Generation systems that produce output similar to the examples in (3) have three kinds of strategies: either a text planning module chooses a discourse connective and decides the position and ordering of clauses (Hovy, 1993) or aggregation is considered to be one of the tasks of the sentence planning module (Shaw, 2002); or a revision module performs aggregation opportunistically (Robin, 1994; Callaway and Lester, 1997). However, none of these systems handle parenthetical constructions in a principled way. Systems where aggregation is part of the text planning module only produce complex sentences made up of clauses joined by discourse connectives – sentence-medial subordinate clauses are not generated at all. In revision-based systems, the output often needs to be corrected after aggregation. For example, Robin's system includes various transformations to correct redundancies, ambiguities or invalid lexical collocations introduced by the revision module. In Shaw's system, the referring expression generation module is run twice, once before and once after aggregation. In general, the ordering of aggregation rules and the interactions between them pose further problems where aggregation is separated into an independent module.We propose a different approach to modeling interactions between linguistic information in separate grammar modules. We argue that constraints that are at the interface of modules (syntactic constraints on referring expressions, discourse-level constraints on syntax, constraints imposed by layout on discourse, etc.) should be stored in an integrated grammar, and only straightforward decisions — which do not require information from a separate grammatical level — should be separated out into individual modules.

As an example, we show a grammar which is capable of generating parenthetical constructions in a principled way. The grammar includes

- a representation for discourse connectives and discourse-level constraints they impose on syntax;
- referring expressions and syntactic constraints on them;

- elements of layout (punctuation marks for main clauses and parentheticals).

We show that by incorporating the above kinds of linguistic information into the grammar of a surface realizer we can improve the flexibility of the system (i.e., generate more paraphrases for the same input) and improve the quality of the generated text without adding more modules to the system.

2.1 Syntactic constraints on pronominalization

To design a grammar for parenthetical constructions, we have carried out a corpus study on embedded rhetorical relations in the RST treebank (Banik and Lee, 2008). The corpus study has shown that the most numerous class of embedded subordinate clauses that occur in sentence-medial position contain a subject pronoun (as in 4a). This embedded subject pronoun in all cases referred back to the subject of the matrix clause, which always immediately preceded the subordinate clause. The pronoun can be either explicit (as in 4a) or implicit (as in the examples in 1). Of the 119 sentence-medial subordinate clauses that we looked at in the study, 35 were of this type (what we call pseudo-relatives).[1] This suggests that in sentence-medial subordinate clauses (or sentence-final ones immediately following the main clause object) the type of a referring expression is solely determined by syntax, much like a WH-pronoun in relative clauses.

(4) a Elixir, since it contains Gestodene, is banned by the FDA.

b # Elixir, since Elixir contains Gestodene, is banned by the FDA.

c # It, since Elixir contains Gestodene, is banned by the FDA.

d # The FDA, since it contains Gestodene, banned Elixir.

The constraints on the form of referring expressions selected for the matrix clause and subordinate clause subjects in these cases can be stated as follows:

- the subject of the subordinate clause has to be realized as a pronoun. (c.f. 4b)
- the subject of the main clause cannot be a pronoun (c.f. 4c)
- the subject pronoun in the subordinate clause will be resolved as referring to an entity mentioned in the matrix clause; this entity has to precede the subordinate clause (c.f. 4d)

In addition to modeling the above constraints, in order to generate parentheticals a generation system also has to

- insert the appropriate discourse connective for the subordinate clause (c.f.2b) and
- insert appropriate punctuation marks on either side of the subordinate clause to avoid potential garden path effects.

3 An integrated discourse-syntax grammar

In order to generate coherent discourse, a generation system needs access to a grammar that is capable of representing multisentential text. In modular systems this is typically achieved by two modules: a text planning module which constructs a text plan and a surface realizer that converts the text plan into sentences. However, text planning and linguistic realization are not two independent processes and many linguistic decisions are in fact made by the text planner. The interactions between text planning and linguistic realization in modular systems have been handled in several ways, including backtracking (Appelt, 1985), interleaving the two components (McDonald, 1983) and restrictive planning (Hovy, 1988). These approaches however make the system inflexible because all possible interactions between modules have to be anticipated by the system designer.

Another, more recent approach to tackle this problem is to use lexicalization not only for sentences but also for texts. The theoretical background for lexicalization on the discourse level has been laid down for Tree Adjoining Grammar (Joshi and Schabes, 1997) by several researchers, including Webber (2004), and Danlos (2000). In particular, Danlos (2000) shows that extending lexicalization to the discourse level makes it possible to completely integrate text planning and surface realization.

[1] Of the rest, 30 were 'free' subordinate clauses (subordinate clauses that are equally felicitous in sentence-initial or sentence final positions, typically they do not contain any pronouns). The rest of the cases were either time adverbials (20) or scopal elements (22).

We have designed a Tree Adjoining Grammar for parenthetical constructions following this latter approach. Elementary trees in the grammar are associated with a flat semantic representation. The trees integrate syntax and discourse representations in the sense that each sentence-level elementary tree includes 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 tree) here projects a discourse clause (i.e., a Dc rooted tree). Every predicate that projects a discourse clause is assigned two kinds of elementary trees: a discourse initial tree (e.g., Fig. 1a) and a discourse continuing tree (e.g., Fig. 1b), which takes the preceding discourse clause as an argument.

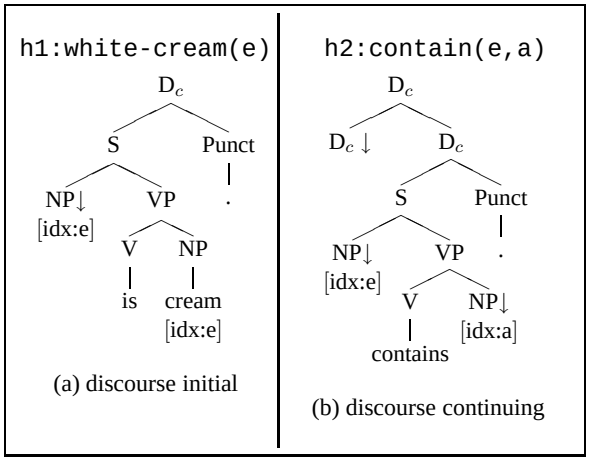


Figure 1: Elementary syntax/discourse trees

The combination of these two trees corresponds to the empty connective ($\oplus$ in Danlos (2000)). Other types of discourse connectives are implemented in the grammar the usual way (see e.g. Danlos (2000)).

4 Referring expressions

One of the challenges of generating paraphrases from a semantic representation is that in some versions there will be a mismatch between the number of noun phrases needed to make the output syntactically well-formed and the number of semantic arguments in the input which can potentially become a noun phrase.

This happens whenever a discourse entity is the argument of more than one semantic predicate. For example, (5) shows possible realizations of the following input where (5a) contains three syntactic slots for "Elixir", (5b,c) contain two slots, and (5d) only one:

```
h0:white-cream(e)
h1:contains(e,g)
h2:elixir(e)
h3:gestodene(g)
h4:ban(f,e)
h5:fda(f)
```

(5) a Elixir is a white cream. Elixir contains gestodene. Elixir is banned by the FDA.
b This white cream, Elixir, contains gestodene. It is banned by the FDA.
c Elixir is a white cream, which contains gestodene. It is banned by the FDA.
d Elixir, a white cream banned by the FDA, contains gestodene.

The task of a generation system is to decide what predicate-argument structure to choose and to decide how the individual noun phrases should be represented. In most systems creating the syntactic "slots" is the task of a text planning or sentence planning module, and filling them in with the right noun phrases is the task of a referring expression generation module, i.e., the referring expression module decides whether an NP slot should be realized as a name, a pronoun or a description.

This division of labour makes it difficult to represent syntactic constraints on pronominalization exhibited by the examples in the previous section, where pronouns are either prohibited or obligatory in specific syntactic contexts.

To model these constraints we include a representation for underspecified referring expressions in the grammar by replacing NP substitution nodes with a referring expression leaf node as illustrated in Fig.2. This allows syntactic constraints to be 'posted' on referring expressions in the appropriate contexts while completely specifying the form of the underspecified slots still remains the task of a referring expression module. In other words, we factor out pronominalization decisions dictated by syntax from pronominalization decisions dictated by discourse level constraints.

Treating pronouns in subordinate clauses differently from pronouns in main clauses has independent justification from psycholinguistics and theoretical linguistics. For example, Miltsakaki (2003) has carried out psycholinguistic experiments on complex sentences containing relative clauses. The experiments show that pronouns in embedded clauses tend to refer back to an entity

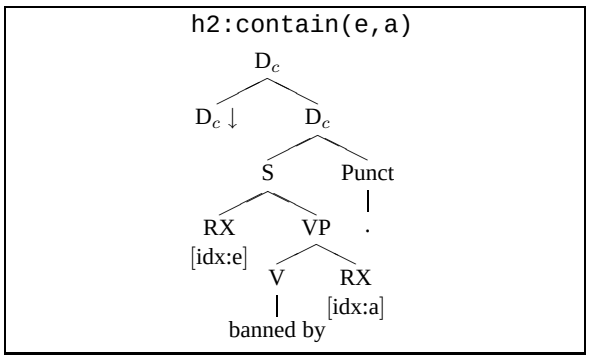


Figure 2: Elementary trees with referring expressions

in the matrix clause, whereas referring expressions in main clauses tend to find their antecedent in the previous main clause. This suggests that pronominalization should be treated differently in subordinate clauses than in main clauses. Research in theoretical linguistics underlines this claim, where Kehler (2002) has shown that apparent discrepancies between different accounts of pronominalization can be reconciled if each method is applied in a different discourse context.

To sum up, in this integrated approach part of the job of the referring expression generation module is taken over by the grammar, namely

- pronominalization of discourse entities in subordinate clauses and
- decisions about when *not* to realize underspecified referring expressions as pronouns.

5 Representing parenthetical constructions

Integrating referring expressions into the grammar this way makes it possible to state syntactic constraints on pronominalization.

5.1 Pronoun prohibited

(6) a Elixir, an illegal drug, is banned by the FDA.

b # It, an illegal drug, is banned by the FDA.

The constraint that parenthetical constructions such as appositives, relative clauses or parenthetical subordinate clauses cannot follow a pronoun is illustrated by the contrast in (6). Using the elementary trees described in the previous section this constraint can now be stated by adding a feature (`[pron:no]`) to the foot node of auxiliary trees, as illustrated in Fig. 3. When the auxiliary tree is adjoined onto an NP, the feature is percolated to the underspecified referring expression node, which will block the referring expression module from realizing this noun phrase as a pronoun.

5.2 Pronoun obligatory

(7) a Elixir, since it contains Gestodene, is banned by the FDA.

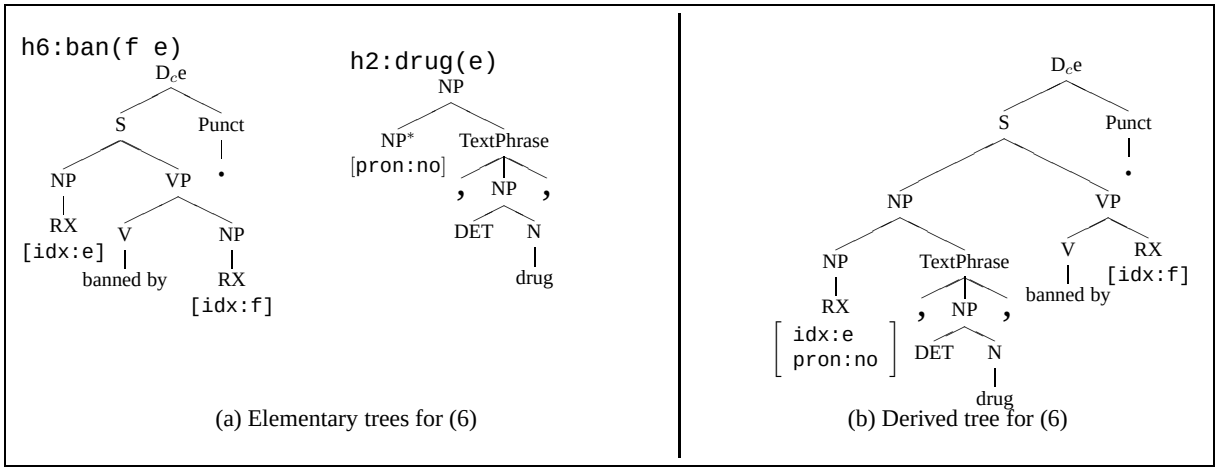
b # Elixir, since Elixir contains Gestodene, is banned by the FDA.

Another case where syntax imposes constraints on pronominalization is contexts where pronouns are not allowed, as illustrated by the example in (7). The discourse connective ‘since‘ is assigned an NP auxiliary tree in this context, which takes the embedded clause as an argument. The features on the auxiliary tree state that the subject of this embedded clause should be expressed by a pronoun and that it should refer to the same discourse entity as the head noun that the auxiliary tree adjoins to. When the discourse connective is combined with the embedded clause, these features are percolated to the referring expression in subject position, requiring it to be realized by a pronoun. Figure 4 illustrates the elementary trees and the derived tree for the embedded clause in (7).

6 Comparison

As an experiment, we have implemented a grammar fragment in the GenI surface realizer (Kow, 2007) and regenerated an example from the ICONOCLAST generator (Power et al., 2003). The example we used is represented by the following input semantics:

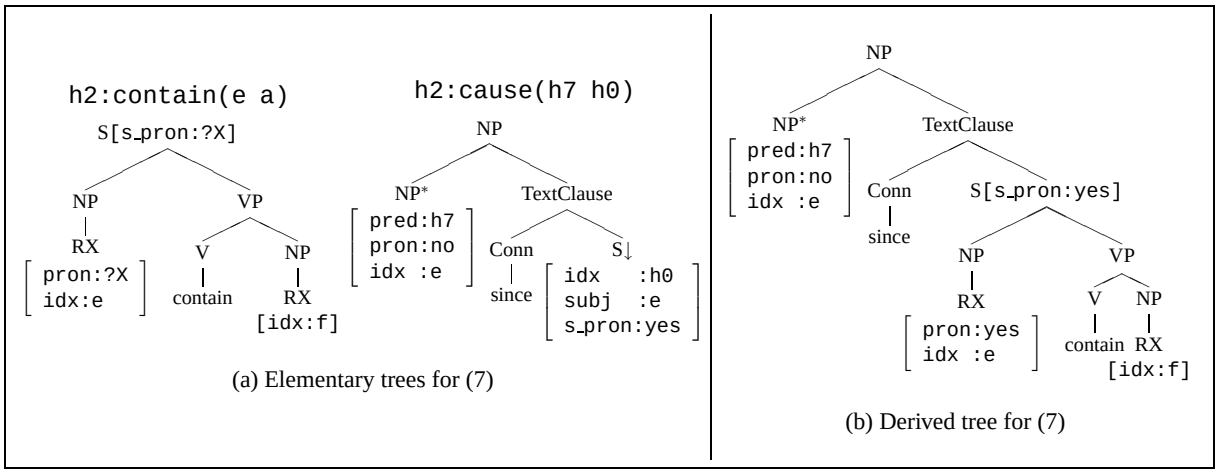
```
h1:   elixir(e)
h2:   fda(f)
h3:   elixir_plus(p)
h4:   gestodene(g)
h5:   contain(e g)
h6:   ban(f e)
h7:   approve(f p)
h8:   concession(h6 h7)
h9:   cause(h5 h6)
h10:  contain(p o)
h11:  oestradiol(o)
h12:  cause(h10 h7)
```



(a) Elementary trees for (6)

(b) Derived tree for (6)

Figure 3: Pronouns not allowed before an appositive



(a) Elementary trees for (7)

(b) Derived tree for (7)

Figure 4: Obligatory pronouns in parenthetical subordinate clauses

ICONOCLAST is a constraint-based system which integrates text planning, document planning and pronominalization to generate all possible paraphrases for a given input. It uses a version of Centering Theory (Grosz et al., 1995) adapted to natural language generation to decide when to pronominalize noun phrases in the generated text. ICONOCLAST has an overgenerate and test approach, where all possible paraphrases are generated and the solutions are ranked according to a set of soft constraints. The system generated 172 solutions for the above input, of which (8) illustrates the top three:

(8) a Since Elixir contains gestodene it is banned by the FDA. However, the FDA approves Elixir Plus since Elixir Plus contains oestradiol.

b Elixir contains gestodene so it is banned by the FDA. However, the FDA approves ElixirPlus since ElixirPlus contains oestradiol.

c Elixir is banned by the FDA since it contains gestodene. However, ElixirPlus is approved by the FDA since it contains oestradiol.

We have regenerated the same text, using only a surface realizer and the grammar described in the previous sections, without a referring expression generation module. A post-processing script transforms RX nodes into a pronoun when they have the relevant feature ([pron:yes]) and into a name when the [pron] feature is missing or its value is no. The surface realizer produced 208 solutions for the same input, of which 96 contained parentheticals. Some of the output is illustrated in (9). Since sentence final parenthetical constructions are impossible to distinguish from sentence-final subordinate clauses in many cases, there is an overlap between the solutions generated by ICONOCLAST and the 96 solutions generated by our

(9) a The FDA bans Elixir since Elixir contains gestodene. However, Elixir Plus (since it contains oestradiol) is approved by the FDA.
b Since Elixir Plus contains oestradiol, although the FDA bans Elixir (since it contains gestodene), Elixir Plus is approved by the FDA.
c Elixir contains gestodene. Consequently, Elixir is banned by the FDA. However, Elixir Plus (since it contains oestradiol) is approved by the FDA.
d Elixir Plus contains oestradiol. Consequently, although the FDA bans Elixir (since it contains gestodene), the FDA approves Elixir Plus.
e Elixir Plus (since it contains oestradiol) is approved by the FDA (although it bans Elixir since it contains gestodene).
f The FDA bans Elixir (since it contains gestodene). However, Elixir Plus is approved by the FDA since Elixir Plus contains oestradiol.

grammar which contain parentheticals. Also, despite the fact that the two systems use the same discourse connectives and a very similar grammar, there are slight differences in the constructions produced. For example, ICONOCLAST allows subordinating conjunctions to "dominate" coordinating conjunctions, producing solutions like the one in (10), although these solutions are assigned at least 4 defects in all cases. These constructions are not allowed in our grammar.

(10) Although Elixir contains gestodene so it is banned by the FDA ElixirPlus contains oestradiol so it is approved by the FDA.

Though comparing the generated solutions is not a straightforward task because of these subtle differences and the sheer number of the solutions produced, the two systems do generate a number of very similar outputs, including the ones shown in (8). However, a significant difference is that our system generates coherent texts which include parenthetical constructions, and which are not generated by ICONOCLAST at all.

7 Related work

Our grammar design was inspired by three discourse-level extensions of Lexicalized Tree Adjoining Grammar. A common idea behind all these approaches is to build an integrated text understanding or generation system in which the same mechanisms are used for the sentence and discourse levels.

DLTAG (Webber, 2004) is an extension of LTAG in which discourse syntax is projected by different types of discourse connectives. In this approach discourse-level syntax is considered to be a separate layer on top of sentence-level syntax and there are two kinds of discourse connectives: anaphoric and structural (Webber et al., 2003). This analysis is not suitable for natural language generation systems which need to have an explicit representation for the arguments of discourse connectives.

G-TAG (Danlos, 2000) is another discourse-level extension of TAG where underspecified 'g-derivation trees' are created for a conceptual input and grouped into lexical databases. A g-derivation tree specifies a set of surface variants, one of which is produced by linearization of the g-derived tree. The other surface variants are created by a post-processing module. While this methodology efficiently reduces the search space of solutions by grouping them together, it assumes that all variants of the same sentence can be generated in the same discourse context.

Most recently, Danlos (2008) introduces D-STAG, a discourse level synchronous TAG coupled with Segmented Discourse Representation Theory (Asher, 1993). In this framework the sentential grammar (S-TAG) and the discourse grammar (D-STAG) are not integrated, therefore discourses where arguments of discourse relations come from discontinuous text spans (as in relative clauses or other types of parentheticals) are not handled by the theory.

8 Conclusions

We have presented an argument against modularizing linguistic information in natural language generation systems. We have argued that complex linguistic constructions which require interactions between several system components are best represented in natural language generation

systems using an integrated grammar. As an example, we have presented the problem of generating parenthetical constructions. Current natural language generation systems either do not generate these constructions at all, or if they do, they do not have a principled approach to the problem and generate parentheticals by adding more modules to a pipeline. We have shown that parentheticals can be generated in a principled way using a surface realizer, when it is equipped with an integrated grammar which incorporates information about syntax, discourse and referring expressions. The solutions produced by our surface realizer demonstrate that this approach enhances the fluency of the generated text and the flexibility of generation systems, without adding extra components or changing the system's architecture.

References

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

N. Asher. 1993. *Reference to Abstract Objects in English*. Kluwer, Dordrecht.

E. Banik and A. Lee. 2008. A study of parentheticals in discourse corpora – implications for NLG systems. In *Proceedings of LREC 2008, Marrakesh*.

C. B. Callaway and J. C. Lester. 1997. Dynamically improving explanations: A revision-based approach to explanation generation. In *Fifteenth International Joint Conference on Artificial Intelligence*, pages 952–58, Nagoya, Japan.

C. B. Callaway. 2003. Integrating discourse markers into a pipelined natural language generation architecture. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 264–271.

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.

L. Danlos. 2008. D-STAG: Parsing discourse with synchronous TAG and SDRT background. In *Proceedings of the Third International Workshop on Constraints in Discourse (CID'2008) Postdam*.

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.

E. H. Hovy. 1988. Two types of planning in language generation. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 179–186, Morristown, NJ, USA. Association for Computational Linguistics.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1-2):341–385.

A. K. Joshi and Y. Schabes. 1997. Tree-Adjoining Grammars. In Rosenberg and Salomaa, editors, *Handbook of Formal Languages and Automata*, volume 3, pages 69–124. Springer-Verlag, Heidelberg.

A. Kehler. 2002. *Coherence, Reference and the Theory of Grammar*. CSLI.

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

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

D. D. McDonald. 1983. Natural language generation as a computational problem. In M. Brady and Robert Berwick, editors, *Computational Models of Discourse*, pages 209–265. MIT Press.

E. Miltsakaki. 2003. *The Syntax-Discourse Interface: Effects of the Main-Subordinate Distinction on Attention Structure*. Ph.D. thesis, Department of Linguistics, University of Pennsylvania.

R. Power N. Bouayad-Agha, D. Scott. 2001. The influence of layout on the interpretation of referring expressions. In L. Degand Y. Bestgen W. Spooren L. van Waes, editor, *Multidisciplinary Approaches to Discourse*, pages 133–141.

R. Power, D. Scott, and N. Bouayad-Agha. 2003. Document structure. *Computational Linguistics*, 29(4):211–260.

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

D. Scott and C. S. Souza. 1990. Getting the message across in RST-based text generation. In C. Mellish R. Dale M. Zock, editor, *Current Research in Natural Language Generation*, pages 31–56. Academic Press.

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

B. Webber, M. Stone, A. Joshi, and A. Knott. 2003. Anaphora and discourse structure. *Computational Linguistics*, 29(4):545–587.

B. Webber. 2004. D-LTAG: extending lexicalized TAG to discourse. *Cognitive Science*, 28(5):751–779.