Segregatory Coordination and Ellipsis in Text Generation

James Shaw
Dept. of Computer Science
Columbia University
New York, NY 10027, USA
shaw@cs.columbia.edu

Abstract

In this paper, we provide an account of how to generate sentences with coordination constructions from clause-sized semantic representations. An algorithm is developed and various examples from linguistic literature will be used to demonstrate that the algorithm does its job well.

1 Introduction

The linguistic literature has described numerous coordination phenomena (Gleitman, 1965; Ross, 1967; Neijt, 1979; Quirk et al., 1985; van Oirsouw, 1987; Steedman, 1990; Pollard and Sag, 1994; Carpenter, 1998). We will not address common problems associated with parsing, such as disambiguation and construction of syntactic structures from a string. Instead, we show how to generate sentences with complex coordinate constructions starting from semantic representations. We have divided the process of generating coordination expressions into two major tasks, identifying recurring elements in the conjoined semantic structure and deleting redundant elements using syntactic information. Using this model, we are able to handle coordination phenomenon uniformly, including difficult cases such as non-constituent coordination.

In this paper, we are specifically interested in the generation of segregatory coordination constructions. In *segregatory* coordination, the coordination of smaller units is logically equivalent to coordination of clauses; for example, "John likes Mary and Nancy" is logically equivalent to "John likes Mary" and "John likes Nancy". Other similar conjunction coordination phenomena, such as *combinatory* and *rhetorical* coordination, are treated differently in text generation systems. Since these constructions cannot be analyzed as separate clauses, we will define them here, but will not describe them further in the paper. In combinatory coordination, the sentence "Mary and Nancy are sisters." is not equivalent to "Mary is a sister." and "Nancy is a sister." The coordinator "and" sometimes can function as a rhetorical marker as in "The train sounded the whistle and [then] departed the station."[1]

To illustrate the common usage of coordination constructions, we will use a system which generates reports describing how much work each employee has performed in an imaginary supermarket human resource department. Generating a separate sentence for each tuple in the relational database would result in: "John rearranged cereals in Aisle 2 on Monday. John rearranged candies in Aisle 2 on Tuesday." A system capable of generating segregatory coordination construction can produce a shorter sentence: "John rearranged cereals in Aisle 2 on Monday and candies on Tuesday."

In the next section, we briefly describe the architecture of our generation system and the modules that handle coordination construction. A comparison with related work in text generation is presented in Section 3. In Section 4, we describe the semantic representation used for coordination. An algorithm for carrying out segregatory coordination is provided in Section 5 with an example. In Section 6, we will analyze various examples taken from linguistic literature and describe how they are handled by the current algorithm.

2 Generation Architecture

Traditional text generation systems contain a strategic and a tactical component. The strategic component determines what to say and the order in which to say it while the tactical component determines how to say it. Even though

[1]The string enclosed in symbols [and] are deleted from the surface expression, but these concepts exist in the semantic representation.

the strategic component must first decide which clauses potentially might be combined, it does not have access to lexical and syntactic knowledge to perform clause combining as the tactical component does. We have implemented a sentence planner, CASPER (**C**lause **A**ggregation in **S**entence **P**lann**ER**), as the first module in the tactical component to handle clause combining. The main tasks of the sentence planner are clause aggregation, sentence boundary determination and paraphrasing decisions based on context (Wanner and Hovy, 1996; Shaw, 1995). The output of the sentence planner is an ordered list of semantic structures each of which can be realized as a sentence. A lexical chooser, based on a lexicon and the preferences specified from the sentence planner, determines the lexical items to represent the semantic concepts in the representation. The lexicalized result is then transformed into a syntactic structure and linearized into a string using FUF/SURGE (Elhadad, 1993; Robin, 1995), a realization component based on Functional Unification Grammar (Halliday, 1994; Kay, 1984).

Though every component in the architecture contributes to the generation of coordinate constructions, most of the coordination actions take place in the sentence planner and the lexical chooser. These two modules reflect the two main tasks of generating coordination conjunction: the sentence planner identifies recurring elements among the coordinated propositions, and the lexical chooser determines which recurring elements to delete. The reason for such a division is that ellipsis depends on the sequential order of the recurring elements at surface level. This information is only available after syntactic and lexical decisions have been made. For example, in "On Monday, John rearranged cereals in Aisle 2 and cookies in Aisle 4.", the second time PP is deleted, but in "John rearranged cereals in Aisle 2 and cookies in Aisle 4 on Monday.", the first time PP is deleted.[2] CASPER only marks the elements as recurring and let the lexical chooser make deletion decisions later. A more detailed description is provided in Section 5.

[2]The expanded first example is "On Monday, John rearranged cereals in Aisle 2 and [on Monday], [John] [rearranged] cookies in Aisle 4." The expanded second example is "John rearranged cereals in Aisle 2 [on Monday] and [John] [rearranged] cookies in Aisle 4 on Monday."

3 Related Work

Because sentences with coordination can express a lot of information with fewer words, many text generation systems have implemented the generation of coordination with various levels of complexities. In earlier systems such as EPICURE (Dale, 1992), sentences with conjunction are formed in the strategic component as discourse-level optimizations. Current systems handle aggregations decisions including coordination and lexical aggregation, such as transforming propositions into modifiers (adjectives, prepositional phrases, or relative clauses), in a sentence planner (Scott and de Souza, 1990; Dalianis and Hovy, 1993; Huang and Fiedler, 1996; Callaway and Lester, 1997; Shaw, 1998). Though other systems have implemented coordination, their aggregation rules only handle simple conjunction inside a syntactic structure, such as subject, object, or predicate. In contrast to these localized rules, the staged algorithm used in CASPER is global in the sense that it tries to find the most concise coordination structures across all the propositions. In addition, a simple heuristic was proposed to avoid generating overly complex and potentially ambiguous sentences as a result of coordination. CASPER also systematically handles ellipsis and coordination in prepositional clauses which were not addressed before. When multiple propositions are combined, the sequential order of the propositions is an interesting issue. (Dalianis and Hovy, 1993) proposed a domain specific ordering, such as preferring a proposition with an animate subject to appear before a proposition with an inanimate subject. CASPER sequentializes the propositions according to an order that allows the most concise coordination of propositions.

4 The Semantic Representation

CASPER uses a representation influenced by Lexical-Functional Grammar (Kaplan and Bresnan, 1982) and Semantic Structures (Jackendoff, 1990). While it would have been natural to use thematic roles proposed in Functional Grammar, because our realization component, FUF/SURGE, uses them, these roles would add more complexity into the coordination process. One major task of generating coordination expression is identifying identical elements in the propositions being combined. In Func-

```
((pred ((pred c-lose) (type EVENT)
        (tense past)))
 (arg1 ((pred c-name) (type THING)
        (first-name ''John'')))
 (arg2 ((pred c-laptop) (type THING)
        (specific no)
        (mod ((pred c-expensive)
              (type ATTRIBUTE)))))
 (mod ((pred c-yesterday)
       (type TIME))))
```

Figure 1: Semantic representation for "John lost an expensive laptop yesterday."

```
Al re-stocked milk in Aisle 5 on Monday.
Al re-stocked coffee in Aisle 2 on Monday.
Al re-stocked tea in Aisle 2 on Monday.
Al re-stocked bread in Aisle 3 on Friday.
```

Figure 2: A sample of input semantic representations in surface form.

tional Grammar, different processes have different names for their thematic roles (e.g., MENTAL process has role SENSER for agent while INTENSIVE process has role IDENTIFIED). As a result, identifying identical elements under various thematic roles requires looking at the process first in order to figure out which thematic roles should be checked for redundancy. Compared to Lexical-Functional Grammar which uses the same feature names, the thematic roles for Functional Grammar makes the identifying task more complicated.

In our representation, the roles for each event or state are PRED, ARG1, ARG2, ARG3, and MOD. The slot PRED stores the verb concept. Depending on the concept in PRED, ARG1, ARG2, and ARG3 can take on different thematic roles, such as Actor, Beneficiary, and Goal in "John gave Mary a red book yesterday." respectively. The optional slot MOD stores modifiers of the PRED. It can have one or multiple circumstantial elements, including MANNER, PLACE, or TIME. Inside each argument slot, it too has a MOD slot to store information such as POSSESSOR or ATTRIBUTE. An example of the semantic representation is provided in Figure 1.

5 Coordination Algorithm

We have divided the algorithm into four stages, where the first three stages take place in the sentence planner and the last stage takes place

```
Al re-stocked coffee in Aisle 2 on Monday.
Al re-stocked tea in Aisle 2 on Monday.
Al re-stocked milk in Aisle 5 on Monday.
Al re-stocked bread in Aisle 3 on Friday.
```

Figure 3: Propositions in surface form after Stage 1.

in the lexical chooser:

Stage 1: group propositions and order them according to their similarities while satisfying pragmatic and contextual constraints.

Stage 2: determine recurring elements in the ordered propositions that will be combined.

Stage 3: create a sentence boundary when the combined clause reaches pre-determined thresholds.

Stage 4: determine which recurring elements are redundant and should be deleted.

In the following sections, we provide detail on each stage. To illustrate, we use the imaginary employee report generation system for a human resource department in a supermarket.

5.1 Group and Order Propositions

It is desirable to group together propositions with similar elements because these elements are likely to be inferable and thus redundant at surface level and deleted. There are many ways to group and order propositions based on similarities. For the propositions in Figure 2, the semantic representations have the following slots: PRED, ARG1, ARG2, MOD-PLACE, and MOD-TIME. To identify which slot has the most similarity among its elements, we calculate the number of distinct elements in each slot across the propositions, which we call NDE (number of distinct elements). For the purpose of generating concise text, the system prefers to group propositions which result in as many slots with NDE = 1 as possible. For the propositions in Figure 2, both NDEs of PRED and ARG1 are 1 because all the actions are "re-stock" and all the agents are "Al"; the NDE for ARG2 is 4 because it contains 4 distinct elements: "milk", "coffee", "tea", and "bread"; similarly, the NDE of MOD-PLACE is 3; the NDE of MOD-TIME is 2 ("on Monday" and "on Friday").

The algorithm re-orders the propositions by sorting the elements in each slots using comparison operators which can determine that Monday is smaller than Tuesday, or Aisle 2 is smaller than Aisle 4. Starting from the slots with largest NDE to the lowest, the algorithm re-

```
((pred c-and) (type LIST)
 (elts
   ~(((pred ((pred "re-stocked") (type EVENT)
             (status RECURRING)))
      (arg1 ((pred "Al") (TYPE THING)
             (status RECURRING)))
      (arg2 ((pred "tea") (type THING)))
      (mod  ((pred "on") (type TIME)
             (arg1 ((pred "Monday")
                    (type TIME-THING))))))
     ((pred ((pred "re-stocked") (type EVENT)
             (status RECURRING)))
      (arg1 ((pred "Al") (TYPE THING)
             (status RECURRING)))
      (arg2 ((pred "milk") (type THING)))
      (mod  ((pred "on") (type TIME)
             (arg1 ((pred "Friday")
                    (type TIME-THING)))))))))
```

Figure 4: The simplified semantic representation for "Al re-stocked tea on Monday and milk on Friday." Note: $\sim() \equiv$ a list.

orders the propositions based on the elements of each particular slot. In this case, propositions will ordered according to their ARG2 first, followed by MOD-PLACE, MOD-TIME, ARG1, and PRED. The sorting process will put similar propositions adjacent to each other as shown in Figure 3.

5.2 Identify Recurring Elements

The current algorithm makes its decisions in a sequential order and it combines only two propositions at any one time. The *result proposition* is a semantic representation which represents the result of combining the propositions. One task of the sentence planner is to find a way to combine the next proposition in the ordered propositions into the resulting proposition. In Stage 2, it is concerned with how many slots have distinct values and which slots they are. When multiple adjacent propositions have only one slot with distinct elements, these propositions are *1-distinct*. A special optimization can be carried out between the 1-distinct propositions by conjoining their distinct elements into a coordinate structure, such as conjoined verbs, nouns, or adjectives. McCawley (McCawley, 1981) described this phenomenon as *Conjunction Reduction* – "whereby conjoined clauses that differ only in one item can be replaced by a simple clause that involves conjoining that item." In our example, the first and second propositions are 1-distinct at ARG2, and they are combined into a semantic structure representing "Al re-stocked *coffee* and *tea* in Aisle 2 on Monday." If the third proposition is 1-distinct at ARG2 in respect to the result proposition also, the element "milk" in ARG2 of the third proposition would be similarly combined. In the example, it is not. As a result, we cannot combine the third proposition using only conjunction within a syntactic structure.

When the next proposition and the result proposition have more than one distinct slot or their 1-distinct slot is different from the previous 1-distinct slot, the two propositions are said to be *multiple-distinct*. Our approach in combining multiple-distinct propositions is different from previous linguistic analysis. Instead of removing recurring entities right away based on transformation or substitution, the current system generates *every* conjoined multiple-distinct proposition. During the generation process of each conjoined clause, the recurring elements might be prevented from appearing at the surface level because the lexical chooser prevented the realization component from generating any string for such redundant elements. Our multiple-distinct coordination produces what linguistics describes as ellipsis and gapping. Figure 4 shows the result combining two propositions that will result in "Al re-stocked tea on Monday and milk on Friday." Some readers might notice that PRED and ARG1 in both propositions are marked as RECURRING but only subsequent recurring elements are deleted at surface level. The reason will be explained in Section 5.4.

5.3 Determine Sentence Boundary

Unless combining the next proposition into the result proposition will exceed the predetermined parameters for the complexity of a sentence, the algorithm will keep on combining more propositions into the result proposition using 1-distinct or multiple-distinct coordination. In normal cases, the predefined parameter limits the number of propositions conjoined by multiple-distinct coordination to two. In special cases where the same slots across multiple propositions are multiple-distinct, the predetermined limit is ignored. By taking advantage of parallel structures, these propositions can be combined using multiple-distinct procedures without making the coordinate structure more difficult to understand. For example, the sentence "John took aspirin on Monday, peni-

cillin on Tuesday, and Tylenol on Wednesday." is long but quite understandable. Similarly, conjoining a long list of 3-distinct propositions produces understandable sentences too: "John played tennis on Monday, drove to school on Tuesday, and won the lottery on Wednesday." These constraints allow CASPER to produce sentences that are complex and contain a lot of information, but they are also reasonably easy to understand.

5.4 Delete Redundant Elements

Stage 4 handles ellipsis, one of the most difficult phenomena to handle in syntax. In the previous stages, elements that occur more than once among the propositions are marked as RECURRING, but the actual deletion decisions have not been made because CASPER lacks the necessary information. The importance of the surface sequential order can be demonstrated by the following example. In the sentence "On Monday, Al re-stocked coffee and [on Monday,] [Al] removed rotten milk.", the elements in MOD-TIME delete forward (i.e. the subsequent occurrence of the identical constituent disappears). When MOD-TIME elements are realized at the end of the clause, the same elements in MOD-TIME delete backward (i.e. the antecedent occurrence of the identical constituent disappears): "Al re-stocked coffee [on Monday,] and [Al] removed rotten milk on Monday." Our deletion algorithm is an extension to the Directionality Constraint in (Tai, 1969), which is based on syntactic structure. Instead, our algorithm uses the sequential order of the recurring element for making deletion decisions. In general, if a slot is realized at the front or medial of a clause, the recurring elements in that slot delete forward. In the first example, MOD-TIME is realized as the front adverbial while ARG1, "Al", appears in the middle of the clause, so elements in both slots delete forward. On the other hand, if a slot is realized at the end position of a clause, the recurring elements in such slot delete backward, as the MOD-TIME in second example. The extended directionality constraint also applies to conjoined premodifiers and postmodifiers as well, as demonstrated by "in Aisle 3 and [in Aisle] 4", and "at 3 [PM] and [at] 9 PM".

Using the algorithm just described, the result of the supermarket example is concise and easily understandable: "Al re-stocked coffee and tea in Aisle 2 and milk in Aisle 5 on Monday. Al re-stocked bread in Aisle 3 on Friday." Further discourse processing will replace the second "Al" with a pronoun "he", and the adverbial "also" may be inserted too.

1. The Base Plan called for one new fiber activation at CSA 1061 in 1995 Q2.
2. New 150mb_mux multiplexor placements were projected at CSA 1160 and 1335 in 1995 Q2.
3. New 150mb_mux multiplexors were placed at CSA 1178 in 1995 Q4 and at CSA 1835 in 1997 Q1.
4. New 150mb_mux multiplexor placements were projected at CSA 1160, 1335 and 1338 and one new 200mb_mux multiplexor placement at CSA 1913b in 1995 Q2.
5. At CSA 2113, the Base Plan called for 32 working-pair transfers in 1997 Q1 and four working-pair transfers in 1997 Q2 and Q3.

Figure 5: Text generated by CASPER.

CASPER has been used in an upgraded version of PLANDOC(McKeown et al., 1994), a robust, deployed system which generates reports for justifying the cost to the management in telecommunications domain. Some of the current output is shown in Figure 5. In the figure, "CSA" is a location; "Q1" stands for first quarter; "multiplexor" and "working-pair transfer" are telecommunications equipment. The first example is a typical simple proposition in the domain, which consists of PRED, ARG1, ARG2, MOD-PLACE, and MOD-TIME. The second example shows 1-distinct coordination at MOD-PLACE, where the second CSA been deleted. The third example demonstrates coordination of two propositions with multiple-distinct in MOD-PLACE and MOD-TIME. The fourth example shows multiple things: the ARG1 became plural in the first proposition because multiple placements occurred as indicated by simple conjunction in MOD-PLACE; the gapping of the PRED "was projected" in the second clause was based on multiple-distinct coordination. The last example demonstrates the deletion of MOD-PLACE in the second proposition because it is located at the front of the clause at surface level, so MOD-PLACE deletes forward.

6 Linguistic Phenomenon

In this section, we take examples from various linguistic literature (Quirk et al., 1985; van Oir-

souw, 1987) and show how the algorithm developed in Section 5 generates them. We also show how the algorithm can generate sentences with non-constituent coordination, which pose difficulties for most syntactic theories.

Coordination involves elements of equal syntactic status. Linguists have categorized coordination into *simple* and *complex*. Simple coordination conjoins single clauses or clause constituents while complex coordination involves multiple constituents. For example, the coordinate construction in "John *finished his work* and [John] *went home.*" could be viewed as a single proposition containing two coordinate VPs. Based on our algorithm, the phenomenon would be classified as a multiple-distinct coordination between two clauses with deleted ARG1, "John", in the second clause. In our algorithm, the 1-distinct procedure can generate many simple coordinations, including coordinate verbs, nouns, adjectives, PPs, etc. With simple extensions to the algorithm, clauses with relative clauses could be combined and coordinated too.

Complex coordinations involving ellipsis and gapping are much more challenging. In multiple-distinct coordination, each conjoined clause is generated, but recurring elements among the propositions are deleted depending on the extended directionality constraints mentioned in Subsection 5.4. It works because it takes advantage of the parallel structure at the surface level.

Van Oirsouw (van Oirsouw, 1987), based on the literature on coordinate deletion, identified a number of rules which result in deletion under identity: Gapping, which deletes medial material; Right-Node-Raising (RNR), which deletes identical right most constituents in a syntactic tree; VP-deletion (VPD), which deletes identical verbs and handles post-auxiliary deletion (Sag, 1976). Conjunction Reduction (CR), which deletes identical right-most or leftmost material. He pointed out that these four rules reduce the length of a coordination by deleting identical material, and they serve no other purpose. We will describe how our algorithm handles the examples van Oirsouw used in Figure 6.

The algorithm described in Section 5 can use the multiple-distinct procedure to handle all the cases except VPD. In the gapping example, the PRED deletes forward. In RNR, ARG2 deletes backward because it is positioned at the end of the clause. In CR1, even though the medial slot ARG2 should delete forward, it deletes backward because it is considered at the end position of a clause. In this case, once ARG3 (the BENEFICIARY "to Sue") deletes backward, ARG2 is at the end position of a clause. This process does require more intelligent processing in the lexical chooser, but it is not difficult. In CR2, it is straight forward to delete forward because both ARG1 and PRED are medial. The current algorithm does not address VPD. For such a sentence, the system would have generated "John and Peter slept" using 1-distinct.

Gapping: John ate fish and Bill ϕ rice.
RNR: John caught ϕ, and Mary killed the rabid dog.
VPD: John sleeps, and Peter does ϕ, too.
CR1: John gave ϕ ϕ, and Peter sold a record to Sue.
CR2: John gave a book to Mary and ϕ ϕ a record to Sue.

Figure 6: Four coordination rules for identity deletion described by van Oirsouw.

Non-constituent coordination phenomena, the coordination of elements that are not of equal syntactic status, are challenging for syntactic theories. The following non-constituent coordination can be explained nicely with the multiple-distinct procedure. In the sentence, "The spy *was in his forties, of average build,* and *spoke with a slightly foreign accent.*", the coordinated constituents are VP, PP, and VP. Based on our analysis, the sentence could be generated by combining the first two clauses using the 1-distinct procedure, and the third clause is combined using the multiple-distinct procedure, with ARG1 ("the spy") deleted forward.

> The spy was in his forties, [the spy] [was] of average build, and [the spy] spoke with a slightly foreign accent.

7 Conclusion

By separating the generation of coordination constructions into two tasks – identifying recurring elements and deleting redundant elements based on the extended directionality constraints, we are able to handle many coordination constructions correctly, including non-constituent coordinations. Through numerous

examples, we have shown how our algorithm can generate complex coordinate constructions from clause-sized semantic representations. Both the representation and the algorithm have been implemented and used in two different text generation systems (McKeown et al., 1994; McKeown et al., 1997).

8 Acknowledgments

This work is supported by DARPA Contract DAAL01-94-K-0119, the Columbia University Center for Advanced Technology in High Performance Computing and Communications in Healthcare (funded by the New York State Science and Technology Foundation) and NSF Grants GER-90-2406.

References

Charles B. Callaway and James C. Lester. 1997. Dynamically improving explanations: A revision-based approach to explanation generation. In *Proc. of the 15th IJCAI*, pages 952–958, Nagoya, Japan.

Bob Carpenter. 1998. Distribution, collection and quantification: A type-logical account. *To appear in Linguistics and Philosophy.*

Robert Dale. 1992. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes.* MIT Press, Cambridge, MA.

Hercules Dalianis and Eduard Hovy. 1993. Aggregation in natural language generation. In *Proc. of the 4th European Workshop on Natural Language Generation*, Pisa, Italy.

Michael Elhadad. 1993. *Using argumentation to control lexical choice: A functional unification-based approach.* Ph.D. thesis, Columbia University.

Lila R. Gleitman. 1965. Coordinating conjunctions in English. *Language*, 41:260–293.

Michael A. K. Halliday. 1994. *An Introduction to Functional Grammar.* Edward Arnold, London, 2nd edition.

Xiaoron Huang and Armin Fiedler. 1996. Paraphrasing and aggregating argumentative text using text structure. In *Proc. of the 8th International Natural Language Generation Workshop*, pages 21–3, Sussex, UK.

Ray Jackendoff. 1990. *Semantic Structures.* MIT Press, Cambridge, MA.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, chapter 4. MIT Press.

Martin Kay. 1984. Functional Unification Grammar: A formalism for machine translation. In *Proc. of the 10th COLING and 22nd ACL*, pages 75–78.

James D. McCawley. 1981. *Everything that linguists have always wanted to know about logic (but were ashamed to ask).* University of Chicago Press.

Kathleen McKeown, Karen Kukich, and James Shaw. 1994. Practical issues in automatic documentation generation. In *Proc. of the 4th ACL Conference on Applied Natural Language Processing*, pages 7–14, Stuttgart.

Kathleen McKeown, Shimei Pan, James Shaw, Desmond Jordan, and Barry Allen. 1997. Language generation for multimedia healthcare briefings. In *Proc. of the Fifth ACL Conf. on ANLP*, pages 277–282.

Anneke H. Neijt. 1979. *Gapping: a constribution to Sentence Grammar.* Dordrecht: Foris Publications.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago.

Randolph Quirk, Sidney Greebaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language.* Longman Publishers, London.

Jacques Robin. 1995. *Revision-Based Generation of Natural Language Summaries Providing Historical Background.* Ph.D. thesis, Columbia University.

John Robert Ross. 1967. *Constraints on variables in syntax.* Ph.D. thesis, MIT.

Ivan A. Sag. 1976. *Deletion and Logical Form.* Ph.D. thesis, MIT.

Donia R. Scott and Clarisse S. de Souza. 1990. Getting the message across in RST-based text generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, New York.

James Shaw. 1995. Conciseness through aggregation in text generation. In *Proc. of the 33rd ACL (Student Session)*, pages 329–331.

James Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proc. of the 9th International Workshop on Natural Language Generation.*

Mark Steedman. 1990. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–264.

J. H.-Y. Tai. 1969. *Coordination Reduction.* Ph.D. thesis, Indiana University.

Robert van Oirsouw. 1987. *The Syntax of Coordination.* Croom Helm, Beckenham.

Leo Wanner and Eduard Hovy. 1996. The HealthDoc sentence planner. In *Proc. of the 8th International Natural Language Generation Workshop*, pages 1–10, Sussex, UK.