Clause Aggregation Using Linguistic Knowledge

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

Abstract

By combining multiple clauses into one single sentence, a text generation system can express the same amount of information in fewer words and at the same time, produce a great variety of complex constructions. In this paper, we describe hypotactic and paratactic operators for generating complex sentences from clause-sized semantic representations. These two types of operators are portable and reusable because they are based on general resources such as the lexicon and the grammar.

Published in Proc. of IWNLG, 1998.

1 Introduction

An expression is more *concise* than another expression if it conveys the same amount of information in fewer words. Complex sentences generated by combining clauses are more concise than corresponding simple sentences because multiple references to the recurring entities are removed. For example, clauses like “Jones is a patient” and “Jones has hypertension” can be combined into a more concise sentence “Jones is a hypertensive patient.” To illustrate the common occurrence of such repeated entities in generation, let us take a shipping company's database as an example. Each database tuple being conveyed is transformed into one or multiple propositions or clauses (we use these terms interchangeably throughout the paper). Each proposition refers to a piece of information which usually corresponds to a simple sentence. The database might contain multiple shipments to the same location possibly on the same day. Generating a sentence for each tuple separately would contain repetitive and potentially redundant references to the same location or date. Though we used a relational database as the example, the observation about recurring entities in the input is also valid for other types of input, such as execution traces from expert systems.

Casper (**C**lause **A**ggregation in **S**entence **P**lann**ER**) is a sentence planner which focuses on generating concise sentences. Clause aggregation can happen at three levels: inferential, rhetorical, and linguistic. At the inferential level, user modeling, domain knowledge, and common sense reasoning are used to reduce the number of concepts to convey. Such operations are implemented in the content planner and clauses are combined without consulting lexical resources. Text summarization is an application which uses inferential operators extensively. For example, the two sentences “John hit Mary” and “Mary kicked John” might imply that “John and Mary fought.” To define a set of inferential operators for unrestricted text is beyond the state-of-art. Because it is unlikely that the inferential operators for our domains (medical briefings and telephone network plan descriptions) will be reusable for other applications, we have directed our effort into aggregation operations at other levels. At the rhetorical level, clauses are combined based on their rhetorical relationships [Mann and Thompson, 1986], such as CONTRAST and CONDITION. We will take advantage of such information in future aggregation work. At the linguistic level, lexical and syntactic information are used to combine clauses. In this paper, we concentrate on two types

The patient's past medical history is significant for bladder carcinoma$_1$, status post cystectomy with a urostomy tube insertion$_2$, left nephrolithiasis$_3$, status post surgery$_4$, recurrent syncope$_5$, questionable vagovagal$_6$, a neurological workup was negative$_7$, and the EPS was negative$_8$, abdominal aortic aneurysm approximately 5 cm$_9$, high cholesterol$_{10}$, exertional angina$_{11}$, past tobacco smoker, quit about one year ago$_{12}$.

Figure 1: The sentence with maximum number of propositions in the corpus

of linguistic aggregation operators: hypotactic and paratactic. The term, *hypotaxis*, describes the relation between a dependent element and its dominant element. Hypotactic operators transform one clause into a modifier and attach the modifier to the dominant clause. In contrast, *paratactic* aggregation operators combine clauses together using constructions of equal status, such as coordination.

Casper is used in two separate projects, MAGIC (**M**ultimedia **A**bstract **G**eneration for **I**ntensive **C**are) and PLANDoc, to increase the fluency of the generated text. MAGIC [Dalal et al., 1996, McKeown et al., 1997] automatically generates multimedia briefings to describe the post-operative status of a patient after undergoing Coronary Artery Bypass Graft (CABG) surgery. It uses the existing computerized information infrastructure in the operating rooms at Columbia Presbyterian Medical Center. PLANDoc[Kukich et al., 1994, McKeown et al., 1994] generates English summaries based on somewhat cryptic traces of the interaction between planning engineers and LEIS-PLAN™. It documents the timing, placement and cost of new facilities for routes in telephone networks.

In Section 2, we present a corpus analysis to identify the complexity of the target output in MAGIC. Section 3 describes the semantic representation used in Casper. Details of hypotactic operators are presented in Section 4. Paratactic operators are described in Section 5. Section 6 describes related work.

2 Corpus Analysis

We conducted a corpus analysis to study various styles and types of aggregation. The corpus consists of the first few sentences in the discharge summaries for 54 patients in the medical domain. These sentences describe patients' demographics and medical conditions pertinent to patient care in the Intensive Care Unit. In our study, the first step was to find out how many propositions were combined in each sentence. A proposition is defined as a piece of information that the physician (the speaker) might choose to convey in a stand-alone sentence to the nurses in the Intensive Care Unit (the hearer). For example, a sentence "The patient is a 40 year old female admitted for heart surgery" contains three propositions: "The patient is a female.", "The patient is 40 years old.", and "The patient was admitted for heart surgery."

The small corpus contained 121 sentences with 2262 words. From the 121 sentences, we obtained 418 propositions after manual decomposition, with a maximum 12 propositions in a single sentence as shown in Figure 1. On average, there are 3.5 propositions per sentence. Out of 54 summary sentences (the first sentence in each discharge summary) for each patient, doctors prefer to use prepositional phrases (PPs) ("with aortic stenosis") rather than relative clauses ("who likely has endocarditis...") to insert medical conditions into a sentence (35 occurrences vs. 4). In only two cases, both PPs and relative clauses were used; all others have neither. Our studies revealed the following:

- Physicians produce very complex sentences.
- Coordinate constructions are the most popular aggregation operations, followed by PPs, and then adjectives. Present and past participle clauses are less common; relative clauses are rare.
- These aggregation operations result in long distance dependencies and non-constituent coordinations (conjoining constituents with different syntactic types).

The analysis also indicates that people prefer using linguistic devices that are simpler (e.g., words over phrases over clauses) [Scott and de Souza, 1990, Hovy, 1993].

We encountered sentences from the corpus which could be formulated more concisely. The doctors did very little editing to the discharge summaries. In this respect, the summaries are somewhat similar to speech. As a result, doctors prefer to use more flexible linguistic constructions, such as PPs, instead of producing the most concise sentences. Concepts such as "hypertension" and "diabetes" have both noun and adjective forms. Even though the noun form is longer (it is always used together with other words as in "patient with hypertension", or "patient who has hypertension"), the shorter adjective form ("hypertensive patient") did not appear in the corpus. In only one case, an adjective "obese" is used instead of the PP "with obesity" to indicate medical conditions. Since many medical conditions have no adjective forms, such as "peptic ulcers", the speaker is more likely to use noun forms to group together all medical conditions. In addition, more information can be attached to nouns but not adjectives. In the noun form, the medical condition "diabetes" might be modified in the corpus, as in "type 1 diabetes with extensive end organ damage" and "borderline diabetes". Such flexibility with nouns explains the popularity of its usage over adjectives.

In summary, our analysis shows that a high level of aggregation is typical in the domain. Judging from the number of the PPs in comparison to relative clauses used, clause aggregation using simpler syntactic constituents is preferred. Doctors generate summaries in real-time without examining all the information right in front of them. As a result, they might not generate the most concise sentences. MAGIC, on the other hand, generates text off-line, with all the conveying information available. This would allow MAGIC to generate more concise text by taking advantage of linguistic opportunities.

3 Semantic Representation

CASPER uses a representation influenced by Lexical-Functional Grammar (LFG) [Kaplan and Bresnan, 1982] and Semantic Structures [Jackendoff, 1990]. An example of the semantic representation is provided in Figure 2. 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, Goal, and Beneficiary, respectively, as in "John gave a red book to Mary yesterday." 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 adjectives or PPs.

4 Hypotactic Operators

We will use an example from MAGIC to demonstrate how hypotactic operators work. The surface forms of the propositions from the content planner are shown in Figure 3. In addition to the propositions, the content planner also indicates that the focus of the discourse is "the patient", with an entity-id, `ID1`. CASPER picks the first proposition, *1a*, as the dominant proposition because it contains the focus entity, and it has `C-NAME` entity. Since, the entity in focus should appear as early as possible to provide a context, the proposition *1a* is transformed from "The patient has name - Jones" into the semantic representation for "Jones is a patient". The PRED of the proposition is changed from `C-HAS-ATTRIBUTE` to `C-IS-INSTANCE`, in addition to swapping of ARG1 and ARG2. Each proposition is represented similarly to the one shown in Figure 2. We use the concept `C-HAS-ATTRIBUTE` to denote that the entity in ARG1 has the attribute stored in ARG2. Depending on the lexical properties of the attribute in ARG2, the proposition *1e* in Figure 3, can be realized as "the patient has diabetes$_{\texttt{noun}}$" or "the patient is diabetic$_{\texttt{adj}}$".

```
((pred ((pred c-has-attribute) (type EVENT) (tense present)))
 (arg1 ((pred c-doctor)       (type THING)
        (mod ((pred c-patient) (type THING)
              (modify-type POSSESSOR) (entity-id ID1)))))
 (arg2 ((pred c-name) (type THING)
        (last-name "Smith"))))
```

Figure 2: Semantic representation for *1f*, "The patient's doctor is Smith."

```
 1a. The patient has name - Jones.
 1b. The patient has gender - female.
 1c. The patient has age - 80 year.
 1d. The patient has hypertension.
 1e. The patient has diabetes.
 1f. The patient's doctor has name - Smith.
 1g. The patient is undergoing CABG.
```

Figure 3: input propositions for "Ms. Jones is an 80 year old hypertensive diabetic female patient of Doctor Smith undergoing CABG."

To aggregate two propositions using hypotactic operators, the propositions must share some entities in common. When they do, hypotactic operators try to transform one of the clauses into a modifier. Since the goal is to generate concise text, CASPER prefers transforming a proposition into an adjective if possible, then a PP, a participle clause, and if all else fails, a relative clause. This preference of syntactically simple expressions over more complex ones was also proposed in [Scott and de Souza, 1990]. In the future, we plan to incorporate constraints from the corpus to determine which aggregation operators to apply and in what order.

To transform a proposition into an adjective, a propositions must satisfy the following two preconditions. First, the slot PRED of the proposition being transformed must be `C-HAS-ATTRIBUTE` (the patient *has* age - 80 years). The other requirement is that the ARG2 of the proposition (*age - 80 years*) can be mapped to an adjective, as permitted in the lexicon. Using the algorithm, propositions *1b, 1c, 1d, 1e* can all be transformed into adjectives and attached to proposition *1a* resulting in "Jones is an 80 year old hypertensive diabetic female patient." There are two interesting things to note here. First, because of the PRED of the dominant proposition is `C-IS-INSTANCE`, the transformed modifiers (age, gender, etc) are attached to the ARG2 slot of the dominant proposition ("a patient") instead of ARG1 ("Jones"). Second, the sequential order of the modifiers is not determined yet at this stage. The goal of CASPER is to produce a concise semantic representation for a set of propositions and to guarantee that there is at least one way to express the result in the later generation modules. To guarantee expressibility [Meteer, 1991], CASPER looks ahead into the lexicon, but it does not make detailed lexical decisions for efficiency reasons. The exact lexical and syntactic decisions, including the ordering between modifiers, are made later in the lexical chooser.

Consider another proposition: "the patient has peptic ulcers". This proposition cannot be transformed into an adjective because there is no adjective form for `C-PEPTIC-ULCER` in the lexicon. A proposition can be transformed into a PP with a general preposition "with" if the PRED of the proposition is `C-HAS-ATTRIBUTE` and the concept in its ARG2 can be mapped into a noun phrase. If we apply the PP operator to the proposition, we would have "Jones is an 80 year old hypertensive diabetic female patient with peptic ulcers." CASPER currently uses an ontology which can identify that `C-PEPTIC-ULCER`, `C-HYPERTENSION`, and `C-DIABETES` are all medical disorders and group them together for cohesion. Since all these medical conditions can be mapped to nouns but not to adjectives, they will all be realized as PPs: "Jones is an 80 year old female patient with hypertension, diabetes and peptic ulcers."

```
((pred ((pred c-install) (type EVENT) (tense past)))
 (arg1 ((pred c-name)    (TYPE THING)
        (first-name "Alice")))
 (arg2 ((pred c-MS-Office) (type THING)))
 (mod  ~(((pred "on")  (type TIME)
          (arg1 ((pred "Monday") (type TIME-THING))))
         ((pred "for") (type BENEFICIARY)
          (arg1 ((pred c-name)   (type THING)
                 (first-name "John")))))))
```

Figure 4: The attribute-value pair representation for "Alice installed MS Office for John on Monday." $\sim()$ $\equiv$ a list.

In *1f* in Figure 3, "The patient's doctor has name - Smith", is transformed into a PP ("of Smith"). The POSSESSOR modifier in ARG1, as shown in Figure 2, can be transformed into a PP using *of*-genitive[Quirk et al., 1985]. This phenomenon holds for relationships similar to patient/doctor, such as advisor/advisee, and boss/employee.

All propositions can be transformed into a relative clause of another as long as they share a common entity. In the example, proposition *1g* does not satisfy the preconditions of the previous hypotactic operators. In this case, it is combined as a present participle clause because present participle clause is simpler and shorter. The result of the hypotactic operators is a semantic representation for "Jones is an 80 year old hypertensive diabetic female patient of Smith undergoing CABG."

Similar to parsing long sentences, efficiency is an important issue in generating long and complex sentences. Search space grows exponentially in respect to the length in both cases. CASPER is able to generate complex sentences efficiently because it delays the difficult detailed lexical decisions until absolutely needed. At the sentence planning level, CASPER looks ahead into the lexicon and merges those propositions that satisfy the required lexical constraints. This prevents the lexical chooser from trying to combine incompatible clauses later. By determining sentence boundaries before carrying out detailed lexical decisions, CASPER cuts down the search space of the lexical chooser drastically. In STREAK [Robin, 1995], a generation system which also implements hypotactic aggregation, detailed lexical decisions are made whenever a proposition is aggregated. This is costly because the best lexical decisions for n propositions might not be useful or correct for $n+1$ propositions. The strategy generates impressive complex sentences, but for some complex sentences, STREAK took more than half an hour. Since CASPER does not use detailed lexical information when it makes sentence boundary determination, it traded some optimal aggregation for efficiency. Even though the lexicon is accessed twice in our system, CASPER prunes the search space drastically by delaying expensive detailed lexical decisions after it knows about how many concepts are involved in the desired sentence. Efficiency issues in generation were also addressed in [McDonald et al., 1987, Elhadad et al., 1997].

5 Paratactic Operators

We will use an imaginary human resource report system for a technical support team as an example to illustrate our paratactic algorithm. The example shown in Figure 4 has the following slots: PRED, ARG1, ARG2, MOD-BENEFICIARY, MOD-TIME. We currently have two approaches to combine propositions using coordinate constructions. In the first approach, adjacent propositions that have only 1 slot containing distinct elements are collapsed into one proposition with one conjoined slot containing the distinct elements. For example, the following sentence is the result of collapsing two propositions with distinct elements in their MOD-BENEFICIARY slot: "Alice installed Quicken for *Mary* and *Peter* on Tuesday." [McCawley, 1981] described this phenomenon as *Conjunction*

```
Alice installed Excel       for John on Monday.
Bob   removed   WordPerfect for John on Tuesday.
Alice installed Powerpoint  for John on Monday.
Cindy removed   Access      for John on Monday.
```

Figure 5: A sample of input propositions in surface form.

```
Alice installed Excel       for John on Monday.
Alice installed Powerpoint  for John on Monday.
Cindy removed   Access      for John on Monday.
Bob   removed   WordPerfect for John on Tuesday.
```

Figure 6: The propositions in surface form after Stage 1.

Reduction. In the second approach, the conjoined propositions have distinct elements in more than one slot. To combine them, each conjoined proposition is generated, but deletion rules (described later in Section 5.4) are used to ensure the resulting sentence has the correct ellipsis. In the following sentence, the two propositions are distinct at both PRED and ARG2: "John *finished his work* and [John] *went home.*"[1] The ARG1 in second proposition "John" is deleted.

Due to limited space, we only describe the algorithm used in CASPER to produce sentences with coordinations. For a more detailed discussion with relevant linguistic motivations, please see [Shaw, 1998]. 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 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.

We will go into detail of each stage in the following 4 sections.

5.1 Group and Order Propositions

Coordination allows the deletion of recurring entities at the surface level, but only if they are *adjacent*; that is, the propositions containing the entities are sequentially next to each other. As a result, the sequential order of the propositions being coordinated affects the length of the output text. In Step 1, CASPER sequentializes the propositions to allow the maximum number of adjacent recurring entities to produce concise text.

For the proposition in Figure 5, the semantic representations have the following slots: PRED, ARG1, ARG2, MOD-BENEFICIARY, and MOD-TIME. To identify which slot has the most similarity among its elements, we calculate the number of distinct elements (NDE) in each slot across the propositions. For the purpose of generating concise text, CASPER prefers to group propositions which result in as many slots with NDE = 1 as possible. For the propositions in Figure 5, the NDE of MOD-BENEFICIARY is 1 because all the beneficiaries are "John"; the NDEs for both PRED and MOD-TIME are 2 because there are two actions, "install" and "remove", which occurred on either "Monday" or "Tuesday"; the NDE for ARG2 is 4 because it contains "Excel", "WordPerfect", "Powerpoint", and "Access"; similarly, the NDE of ARG1, the agent, is 3.

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

```
((pred c-and) (type LIST)
 (elts
   ~(((pred ((pred "installed") (type EVENT)  (status RECURRING)))
      (arg1 ((pred "Alice")     (TYPE THING)  (status RECURRING)))
      (arg2 ((pred "Excel")     (type THING)))
      (mod  ((pred "on")        (type TIME)
             (arg1 ((pred "Monday") (type TIME-THING))))))
     ((pred ((pred "installed") (type EVENT)  (status RECURRING)))
      (arg1 ((pred "Alice")     (TYPE THING)  (status RECURRING)))
      (arg2 ((pred "Outlook")   (type THING)))
      (mod  ((pred "on")        (type TIME)
             (arg1 ((pred "Friday") (type TIME-THING))))))))))
```

Figure 7: The simplified representation for "Alice installed Excel on Monday and Outlook 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 "Alice" is smaller than "Bob" alphabetically. Starting from the slots with highest NDE to the lowest, the algorithm re-orders the propositions based on the elements of each particular slot. In this case, propositions will ordered according to their ARG2 first, followed by ARG1, MOD-TIME, PRED, and MOD-BENEFICIARY. The sorting process will put similar propositions adjacent to each other as shown in Figure 6.

5.2 Identify Recurring Elements

The current algorithm tries to combine only two propositions at once. In Stage 2, CASPER 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*. Propositions that are 1-distinct can be replaced with one proposition with one slot conjoining the distinct elements of that slot. In our example, the first and second propositions are 1-distinct at ARG2, and they are combined into a semantic structure representing "Alice installed *Excel* and *Powerpoint* for John on Monday."

When propositions 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 immediately based on transformation or substitution, the current system generates *every* conjoined multiple-distinct proposition. During the lexicalization of the conjoined sentence, the lexical chooser prevents the realization component from generating any string for the redundant elements. Our multiple-distinct coordination produces what linguists describe as ellipsis and gapping. Figure 7 shows the result combining two propositions that will result in "Alice installed Excel on Monday and Outlook on Friday." Some readers might notice that PRED and ARG1 in both propositions are marked as RECURRING. The process to delete only subsequent recurring elements at surface level will be explained in Section 5.4.

5.3 Determine Sentence Boundary

Unless combining the next proposition into the result proposition will exceed the pre-determined 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. Based on looking at PLANDOC output, we limit the number of propositions conjoined by multiple-distinct coordination to two in normal cases. Higher threshold renders some of the sentences difficult to comprehend. In special cases where the same slots across multiple propositions are multiple-distinct, the pre-determined 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, penicillin 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 easily understandable complex sentences containing a lot of information.

5.4 Delete Redundant Elements

Stage 4 handles ellipsis. In the previous stages, adjacent 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, Alice installed Excel and [on Monday,] [Alice] removed Lotus 123.”, 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): “Alice installed Excel [on Monday,] and [Alice] removed Lotus 123 on Monday.” 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, “Alice”, 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. Our extended directionality constraint, an extension of [Tai, 1969]'s 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 is concise and easily understandable: “On Monday, Alice installed Excel and Powerpoint and Cindy removed Word for John. Bob removed WordPerfect for John on Tuesday.” Further discourse processing can replace the beneficiary“John” in the second sentence with a pronoun “him”.

6 Related Work

Both hypotactic and paratactic constructions described in this paper have received a lot of attention in linguistics [Quirk et al., 1985, Halliday, 1994, Carpenter, 1998]. Much generation literature on aggregation was disguised under the topic “revision” [Meteer, 1991, Robin, 1995] [Callaway and Lester, 1997]. We consider clause aggregation as an integral part of a text generation system, not as a revision. The term “revision” implies that something has been generated and then improved upon, which is not the case in these systems. We prefer the term *optimization* used by [Dale, 1992], which describes the phenomenon of aggregation more appropriately – it use fewer words to convey the same amount of information.

In earlier systems, clause aggregations are implemented in strategic component [Mann and Moore, 1980, Dale, 1992, Horacek, 1992]. Logical derivations were used to combine clauses and remove easily inferable clauses in [Mann and Moore, 1980]. In such systems, aggregation decisions are made without lexical information. Newer systems, such as [Shaw, 1995, Wanner and Hovy, 1996, Huang and Fiedler, 1997], use a sentence planner to make decisions at clause level between the strategic and tactical component.

With the exception of [Scott and de Souza, 1990] and [Robin, 1995], most research in aggregation did not transform clauses into modifiers, such as adjectives, PP, or relative clauses, in a systematic manner. [Scott and de Souza, 1990] proposed heuristics for carrying out clause combining based on RST and specifically identified which rhetorical relations are appropriate for “embedding”, which corresponds to our hypotactic operators. We will incorporate rhetorical aggregation in the future. Robin's work on revision operators is similar to ours. We have describe his work earlier in Section 4.

Because sentences with coordination constructions can express a lot of information with few words, many text generation systems have implemented the generation of coordination expressions with various complexities [Dale, 1992, Dalianis and Hovy, 1993, Huang and Fiedler, 1997, Shaw, 1995, Callaway and Lester, 1997]. Most systems handles simple coordination which contains only one conjoined syntactic constituents, such as subject, verb, or object. None of them handles ellipsis as CASPER does. CASPER tries to systematically find the most concise way to express the propositions by looking through all the propositions. In contrast, aggregation operators proposed in other work are local and does not handle complex cases. In addition, the possibility of too much information in a sentence has not received much attention. Most research simply ignores this possibility because the input to their sentence planners never exceeds a few clauses.

7 Conclusion

We describe how hypotactic operators combine clauses using lexical information and how paratactic operators produce sentences with coordination. Through the use of look-ahead into the lexicon during the aggregation process to guarantee expressibility and by performing the task of sentence delimitation before lexical choice, the system can generate complex sentences efficiently. Since hypotactic, and paratactic operators are reusable, further speed-up in future generation system development is expected.

8 Acknowledgments

The author would like to thank Kathleen McKeown for her valuable advice and encouragement. 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

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

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

[Dalal et al., 1996] Dalal, M., Feiner, S., McKeown, K., Jordan, D., Allen, B., and alSafadi, Y. MAGIC: An experimental system for generating multimedia briefings about post-bypass patient status. In *Proc. 1996 AMIA Annual Fall Symp*, pages 684–688, Washington, DC.

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

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

[Elhadad et al., 1997] Elhadad, M., McKeown, K., and Robin, J. Floating constraints in lexical choice. *Computational Linguistics*, 23(2):195–239.

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

[Horacek, 1992] Horacek, H. An integrated view of text planning. In *Aspects of Automated Natural Language Generation*, Lecture Notes in Artificial Intelligence, 587, pages 29–44. Springer-Verlag, Berlin.

[Hovy, 1993] Hovy, E. H. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63. Special Issue on NLP.

[Huang and Fiedler, 1997] Huang, X. and Fiedler, A. Proof verbalization as an application of NLG. In *Proc. of the 15th IJCAI*, pages 965–970, Nagoya, Japan.

[Jackendoff, 1990] Jackendoff, R. *Semantic Structures*. MIT Press, Cambridge, MA.

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

[Kukich et al., 1994] Kukich, K., McKeown, K., Shaw, J., Robin, J., Lim, J., Morgan, N., and Phillips, J. User-needs analysis and design methodology for an automated document generator. In Zampolli, A., Calzolari, N., and Palmer, M., editors, *Linguistica Computazionale, Vol. IX-X*, pages 109–115. Kluwer Academic Publishers, Norwell, MA.

[Mann and Moore, 1980] Mann, W. C. and Moore, J. A. Computer as author – results and prospects. Technical Report RR-79-82, USC Information Science Institute, Marina del Rey, CA.

[Mann and Thompson, 1986] Mann, W. C. and Thompson, S. A. Rhetorical structure theory: Description and construction of text structures. Technical Report RS-86-174, USC Information Sciences Institute, Marina Del Rey, CA.

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

[McDonald et al., 1987] McDonald, D. D., Meteer, M. M., and Pustejovsky, J. D. Factors contributing to efficiency in natural language generation. In Kempen, G., editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, NATO ASI Series – 135, pages 159–182. Martinus Nijhoff Publishers, Boston.

[McKeown et al., 1994] McKeown, K., Kukich, K., and Shaw, J. Practical issues in automatic documentation generation. In *Proc. of the 4th ACL Conference on Applied Natural Language Processing*, pages 7–14, Stuttgart.

[McKeown et al., 1997] McKeown, K., Pan, S., Shaw, J., Jordan, D., and Allen, B. Language generation for multimedia healthcare briefings. In *Proc. of the Fifth ACL Conf. on ANLP*, pages 277–282.

[Meteer, 1991] Meteer, M. The implications of revisions for natural language generation. In Paris, C. L., Swartout, W. R., and Mann, W. C., editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 155–178. Kluwer Academic Publishers, Boston.

[Quirk et al., 1985] Quirk, R., Greebaum, S., Leech, G., and Svartvik, J. *A Comprehensive Grammar of the English Language*. Longman Publishers, London.

[Robin, 1995] Robin, J. *Revision-Based Generation of Natural Language Summaries Providing Historical Background*. PhD thesis, Columbia University.

[Scott and de Souza, 1990] Scott, D. R. and de Souza, C. S. Getting the message across in RST-based text generation. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation*, pages 47–73. Academic Press, New York.

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

[Shaw, 1998] Shaw, J. Segregatory coordination and ellipsis in text generation. In *To appear in Proc. of the 17th COLING and the 36th Annual Meeting of the ACL.*

[Tai, 1969] Tai, J. H.-Y. *Coordination Reduction*. PhD thesis, Indiana University.

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