SASSY

SOFTWARE ARCHITECTURE SUPPORT SYSTEM

Microplanner Analysis

Publication History

Date	Who	What Changes
13 March 2021		Initial version.

Table of Contents

1 Introduction

This document describes the analysis for the sentence microplanner, the component responsible for planning paragraphs. This component of SASSY creates the input for the nnatural language generator (NLG) module. Its inputs are the output of a discourse planner.

1.1 Scope

The document aims to collect together the research done in the area of planning sentences for use by natural language generators. It will then put forward some alternatives that might be able to be implemented.

1.2 Overview

It is assumed that the discourse planner will provide an extract from the knowledge database that will equate to a paragraph in the final document.

The output from the microplanner will be a set of specifications for the NLG to use to create sentences.

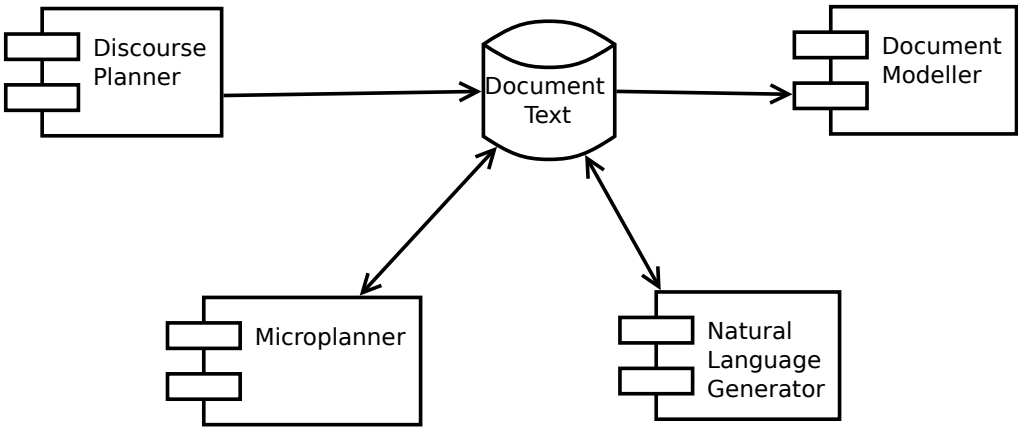
The tasks of the microplanner can be classified as discourse structuring, sentence content deliniation, internal sentence organisation, reference choice and lexical choice. [These terms are defined and expanded in the subsequent sections of the document.]

1.3 Audience

The developers of SASSY.

2 Microplanner Context

This section describes how the microplanner and NLG modules might fit into the SASSY workflow.



The Discourse Planner saves its output in RDF format as the document text. The microplanner then selects sets of RDF statements to work on and saves the resultant NLG specifications. The NLG then processes the specifications into sentences or phrases. Finally the document modeller converts the text into a document model which is then converted to markdown and then PDF (or other formats).

The Document Text is an RDF model containing RDF statements extracted from the knowledge databases by the discourse planner. The microplanner replaces some of the statements with the statements for NLG specifications. The NLG then replaces the specifications with its generated text. The document modeller then selects the document text and creates a new RDF model representing the document.

3 Microplanner Inputs

It is a fundamental tennet of software system design that a software module cannot usefully create new information. Any information not provided for a particular processing instantiation must be available either in reference knowledge databases, or built into the software itself.

The following subsections will be filled out with detail as it becomes clearer what exact information is required by the process.

3.1 Instance Text

This input is the set of RDF statements that the discourse planner has determined will make up a paragraph in the final document. These can be viewed as pure semantic information describing some aspect of a software architecture [or other domain if applicable].

The input will likely be a set of rdf:Statement nodes with its subject, predicate and object predicates indicating the statement in the knowledge base.

It is likely that much of the instance text will include descriptions written by the users of the system. The microplanner should reproduce this text in its output.

3.2 Input Filtering

This filters out some RDF statements that have properties that are not applicable to the document being created. The set of such properties will be an input to the planner. This idea comes from the ONTOSUM[1] system which is designed to summarise an ontology:

> During pre-processing ONTOSUM also removes properties given as values of the propertiesToFilterOut parameter of the Ontology2KBLex module. The function of this parameter is enable the user to exclude some information from the summary. For example, properties encoding the provenance of this instance or providing lexical information (e.g., full-name) may be excluded in this way.

3.3 Reader Model

This will be an RDF model that describes the intended reader of the document. It could include information about the technical expertise and the reading comprehension level, for example. This might also include language and locale. This could then feed into the sentence complexity and the use of technical terminology.

3.4 Document Model

This will be another RDF model. It will describe the style of the document being created. A formal technical specification will use different language from a short summary note, for example.

3.5 Discourse Context

The microplanner will need to know where within the document its output will appear. An abstract or executive summary will be different from the body. For less technical documents there will be a difference between the beginning and the end as terms can be used more freely.

The context might also include information about what the important objects are in the paragraph being created. This should make the document more coherent.

4 Microplanner Outputs

4.1 NLG Input

The main output from the microplanner will be a set of specifications for the NLG module. These specifications will be in the form of RDF statements. There will be a list of specifications that will be translated into a list of sentences by the NLG.

The NLG Specification structure is quite complex with many attributes available for each object. Creating the rules for each of the objects and their attributes will be a very large task.

Expand using the NLG Specification interface. This should be fully documented as part of NLG prior to undertaking this project.

4.2 Document Input

The microplanner will need to inform the document constructor where the generated text needs to go. This could be a normal paragraph, with some level in the document, a section heading, list items, and so forth.

An alternative workflow design might just use the microplanner and the NLG to create a few sentences in the overall document with most of the content taken directly from the descriptions in the knowledge database. In such a case the microplanner would not need to feed directly into the document model.

Expand once we have the document schema.

5 Microplanner Processing

This section provides an overview of the processing that will be required. A more detailed description will follow.

5.1 Reification

Each RDF statement that is selected as part of the input instance text will be converted to its reified form. This will introduce rdf:Statement nodes into the model which are more natural objects to be manipulating.

This step would normally be done by the discourse planning module, however for experimenting and testing it might be useful to have this capability included in the microplanner module as well.

5.2 Concepts

The terms used in the RDF of the instance text input need to be mapped onto terms that have a conceptual meaning that can be used to guide the text generation process.

The natural source for these concept terms is a thesaurus. It is envisaged that the project's data dictionary will include links to a thesaurus entry for each of its terms.

A full thesaurus, such as Roget's, might not be the best solution. A simpler classification system, such as the LOOM Upper Model[2] might be a better alternative. However, we would then have to classify all the terms used.

Another possibility is to link to entries in a lexicon constructed from WordNet and enWiktionary, or an extract of it considering how large that lexicon would be.

A more complete analysis of this should be part of the Lexicon project.

5.3 Discourse Structuring

This is responsible for determining the structure of the paragraph, for example will it have lists, and if so, what type of list.

It will need to determine how to integrate the canned text provided by the users.

It will need to use the discourse context to determine the ordering of clauses within the paragraph. It seems that a discourse should be structured with some **intent**. The discourse must have some purpose and motivation for saying what it is saying.

The module will need to use focus of attention to prevent unnecessary changes of topic.

5.4 Sentence Content Deliniation

This is responsible for aggregating the RDF statements into sentences, or the clauses in lists, for example. It will try to create appropriately complex sentences according to the type of document and the profile of the intended audience.

5.5 Internal Sentence Organisation

This is responsible for things like setting the subject of a clause, and the relative subordination of the clauses.

5.6 Grammar

This is responsible for setting all the grammar flags that the NLG will need.

5.7 Reference Choice

This is responsible for determining how objects are referenced. For example it should determine what can be replaced by pronouns.

5.8 Lexical Choice

When objects don't have a user supplied label, and they have not been referenced previously in the document, they need to be given a suitable label.

This implies that the microplanner needs to be run sequentially over the document, or a separate labelling step is introduced. It also implies there is a lexicon avialable that will allow a label to be chosen based on the semantics of the object.

SASSY will have a lexicon focussed on the software architecture process. However each application that wants to use SASSY will be required to have a suitable lexicon. This is a problem.

> **ONTOSUM[1]** has a module which generates the lexicon from the ontology using labels or a hueristic process that converts property names into labels.

6 Text Generation Literature Review

6.1 A Brief History

It started out with templates, as one would find in an application for mass mail-outs. A few fields are populated from a well known database structure.

This seems to have evolved into "schemas". These are a set of rules which can be applied recursively. Some good results were achieved in the late '80s using this technique[3] However when you read about them in detail they were only useful for extremely restricted data sets and required a lot of input from domain experts to tune them.

Both templates and schemas required very well defined database structures for the input data. They could not handle unexpected data.

In 1987 Mann & Thompson published their "Rhetorical Structure Theory", primarily as a way of analysing text. By 1992 Eduard Hovy had created a text generator using RST. However, it was still very limited in its domain and required a lot of specialised linguistic and domain knowledge to make it work.

In 1997 Daniel Marcu (University of Toronto) pointed out a serious problem with the way RST was being done - it could "finish" without using all the input data. This is not very acceptable if you are trying to get a report of something from a database.

It seems that the academics in text generation had become more interested in creating well formed text, rather than the completeness of its output.

Marcu proposed an algorithm for generating text that used all of the input data. However, as I have just discovered, it proves to be of not much practical use.

From about 2005 text generation seems to have become obsessed with neural network technology. I rather suspect this is a continuation of the effort to get something that reads nicely without much concern for the completeness of the report.

6.2 Discourse Strategies for Generating Natural-Language Text

Kathleen R. McKeown, 1985

Abstract: If a generation system is to produce text in response to a given communicative goal, it must be able to determine what to include in its text and how to organize this information so that it can be easily understood. In this paper, a computational model of discourse strategies is presented that can be used to guide the generation process in its decisions about what to say next. The model is based on an analysis of naturally occurring texts and represents strategies that can be used for three communicative goals: define, compare, and describe. We show how this model has been implemented in TEXT, a system which generates paragraph-length responses to questions about database structure.

This paper[3] describes work done in the early 1980's. It brought together previous research and provides a lot of input to text generation researchers for the next 20 years.

The paper introduces the concept of *focus of attention*.

> Focus of attention constrains the information that needs to be considered when deciding what to say next. It also provides constraints when the discourse strategy allows for several possible choices for what to say next by indicating which information ties in best with the preceding discourse.
>
> ... the tactical component can use the tracking of focus of attention to select, for example, the passive construction over the active.

The paper describes the TEXT system which was designed to answer questions about a naval database. The domain was the database schema, a rather small, static and well defined data set.

The TEXT system uses d*iscourse strategies* to select and order the input data. SASSY will use a separate component to select the data. Ordering may be an issue, but RDF data has natural linkage between the data items that might be leaveraged.

> The basic units of discourse strategies are *rhetorical predicates*. They characterize the predicating acts a speaker may use and delineate the structural relation between propositions in a text.

The rhetorical predicates mentioned here are precursors to those used in Rhetorical Structure Theory[4]. The TEXT system is able to use them to select the data to be incorporated into the output text. (This implies that the system cannot guarantee that a full description of the data will be presented.)

The predicates are arranged into schemata. The contention is that there are a set of recurring patterns that are used in most texts. A schema is represented as a BNF grammar with the predicates as the symbols. The paper identifies four schemata: *identification, constituency, attributive, and contrastive*.

The paper notes that these schemata are not as rigid as a language's grammar:

> All this points to the fact that the schemata do not function as grammars of text. They do, however, identify common means for effectively achieving certain discourse goals. They capture patterns of textual structure that are frequently used by a variety of people (and therefore do not reflect individual variation in style) to successfully communicate information for a particular purpose. Thus, they describe the norm for achieving given discourse goals, although they do not capture all the means for achieving these goals.

For a text generation program this would seem to be adequate. We want a report, not a Shakesperean play.

The paper then posits that the schemata can be used recursively. A schema is defined for each rhetorical predicate in terms of other predicates. This was not done by the researcher - they only did four of the ten predicates identified. The problem gets worse for Rhetorical Structure Theory with 25 predicates, and, I suspect untenable for other systems that have over 50.

> A text generated by applying schemata recursively will be tree-tructured, with a sub-tree occurring at each point where a predicate has been expanded into a schema. Propositions occur at the leaves of the tree.

An important aspect of the process being described is determining what data from the knowledge database is applicable for a rhetorical predicate. The paper is a bit shy about this:

> Each predicate has a function associated with it which 'matches' the predicate against the relevant knowledge pool and returns all propositions in the pool which are classified by the predicate.

It seems likely that these functions are highly tuned to the data. Such an approach is not scalable to the sort of things that SASSY will need to handle.

The recursion process is optional. How deep to go is dependent on external constraints, such as the type of document being constructed and the model of the reader.

> Another issue in the recursive use of schemata is the question of when recursion is necessary. Clearly there are situations where a simple sentence is sufficient for fulfilling a communicative goal, while in other cases, it may be necessary to provide a more detailed explanation.

A large part of the paper is devoted to describing how the TEXT system uses an augmented transition network to construct its output.

Section 6 of the paper discusses Focus of Attention.

> Sidner[5] showed that speakers can either maintain their current focus, shift focus to an item just introduced, return to a previous focus, or focus on an item implicitly related to the current focus.
>
> A preference ordering on Sidner's constraints was developed for generation The ordering suggests that a speaker should shift to focus on an item just

introduced into conversation if possible. If the speaker chooses not to do so, that item will have to be re-introduced into conversation at a later point before the additional information can be conveyed. If, on the other hand, the speaker does shift to the item just mentioned, there will be no trouble in continuing with the old conversation by returning to a previous focus.

The second preference indicates that a speaker should continue talking about the same thing rather than returning to an earlier topic of conversation where possible. By returning to a previous discussion, a speaker closes the current topic. Therefore, having introduced a topic (which may entail the introduction of other topics), one should say all that needs to be said before returning to an earlier topic.

6.3 Rhetorical Structure Theory: A Theory of Text Organisation

William C. Mann, Sandra A. Thompson, 1987

Rhetorical Structure Theory is a descriptive theory of a major aspect of the organization of natural text. It is a linguistically useful method for describing natural texts, characterizing their structure primarily in terms of relations that hold between parts of the text. This paper establishes a new definitional foundation for RST. Definitions are made more systematic and explicit, they introduce a new functional element, and incidentally reflect more experience in text analysis. Along with the definitions, the paper examines three claims and findings of RST: the predominance of nucleus/satellite structural patterns, the functional basis of hierarchy, and the communicative role of text structure.

This[4] is one of the most influential papers in the text generation and analysis categories. While the paper only defines it for analysis, RST formed the theoretical basis of text generation until the focus changed to neural networks.

Please refer to Appendices A and B for details of RST.

It is important to note that the rhetorical relations have little relationship to the semantic predicates within the clauses that make up the text spans. It is also interesting that different analysts will create different structures, indicating that the theory is not very prescriptive.

6.4 Automated Discourse Generation using Discourse Structure Relations

Eduard Hovy, 1993

Abstract: This paper summarizes work over the past five years on the automated planning and generation of multisentence text using discourse structure relations, placing it in context of ongoing efforts by Computational Linguists and Linguists to understand the structure of discourse. Based on a series of studies by the author and others, the paper describes how the orientation of generation toward communicative intentions illuminates the central structural role played by intersegment discourse relations. It outlines several facets of discourse structure relations as they are required by and used in text planners--their nature, and extension to associated tasks such as sentence planning and text formatting.

The paper[6] provides its own summary:

> This paper focuses on discourse structure and discourse structure relations as seen from the text planning perspective. It can serve as a survey of what has been done recently and a pointer to where research can fruitfully be performed. After arguing in Section 2 that without an understanding of discourse structure, communication is unlikely to succeed, the paper outlines various theories of discourse structure, linguistic and computational. Section 3 describes an early computational attempt, the first of several similar efforts, to plan discourse structure automatically by dynamically constructing a tree of interclause operators or relations. These attempts' general requirements for discourse structure are summarized in Section 4. Section 5 then presents four primary aspects of discourse structure relations that arise, regardless of particular theory of discourse structure, when they are employed to plan discourse automatically. Finally, Section 6 describes the effects of discourse structure relations on related tasks such as sentence planning and text formatting.

The problem the paper is trying to solve can be summed up as:

> Of the n! permutations possible for n sentences, usually only a handful of them make semantic sense, and often their meanings differ quite radically.
>
> Several discourse phenomena signal discourse structure, including clause juxtapositioning, pronoun and other reference use, quantifier scoping, focus shifts, tense, and aspect.
>
> Determining the interactions due to sentence juxtaposition can be a significant problem. Unfortunately, there are no grammars of paragraph structure, no general linguistic theories of the parts of speech of discourse and inference. But people do assemble sentences into well-structured multisentence texts in a principled way. What principles do they use? How do the principles relate to inferences? What basic elements govern discourse structure?
>
> The key insight for solving these questions is the notion of text coherence.
>
> ... a discourse is coherent and will succeed only if it is properly structured: if

> (i) segments properly reflect communicative intentions, and (ii) interrelationships among segments are properly expressed, enabling the hearer to recognize them, draw the appropriate inferences, and build up the desired structures.

Section 3 describes an attempt to build a text structurer:

> The first experiment in dynamic text structure planning involved developing a paragraph structure planner [using RST]

> The hardest task in developing the structurer was understanding how to operationalize RST relations. Simultaneously, they had to enforce coherence by capturing the desired hearer inferences, expressing the speaker's communicative goals, and guiding the planning process.

There are some quite troubling aspects to what was done:

> In the structurer's operationalized relations, then, each relation/plan has two primary parts, a *Nucleus* and a *Satellite*, and recursively relates some unit(s) of the input, or another relation (cast as Nucleus), to other unit(s) of the input or another relation (cast as Satellite).

> ... since the Nucleus and Satellite material is usually expanded upon in typical domain-specific ways [...] possible paths of expansion are contained in *growth points*: collections of goals that suggest the inclusion of additional material in appropriate places in the text. Determining the contents of growth points is a major task; in the example Navy domain, for instance, not only were dozens of paragraphs analyzed, but the Navy expert responsible for producing them was interviewed and taped over a period of three days.

This does not bode well for the general purpose requirements of SASSY. While we might be able to do this for the built-in aspects of SASSY itself, it is too much to ask a user of SASSY to do this for their own projects.

> The planning process bottoms out when either all of the input entities have been incorporated into the tree or no extant goals can be satisfied by the remaining input entities. The tree is then traversed in a depth-first left-to-right manner, adding the relations' characteristic cue words or phrases to the appropriate input entities and appropriate syntactic constraints on realization, and transmitting them to Penman to be generated as English sentences.

Note that some input can be ignored. This is problematic for a system that is supposed to support designing a software system. [Perhaps one can arrange for some default goals that suck up all the input?] Also, it is common practice for the first sentence of a paragraph to act as a summarising introduction, which might require that a few of the input entities be reused.

Section 4 provides a set of general requirements for discourse planners.

> No existing theory or description, RST included, has enough descriptive power to support all the needs of text planners. Whether formalist or

functionalist, each theory addresses some phenomena better than others. From the rather specific perspective of text planning, however, the descriptions of discourse used by various text planners are quite similar, a fortunate fact that enables one to synthesize a relatively neutral working definition.

1. Discourse: A discourse (a text) is a structured collection of clauses. By their semantic relatedness, clauses are grouped into segments; the discourse structure is expressed by the nesting of segments within each other according to specific relationships. A discourse can thus be represented as a tree structure, in which each node of the tree governs the segment (subtree) beneath it. At the top level, the discourse is governed by a single root node; at the leaves, the basic segments are single grammatical clauses.

2. Purpose: Each discourse segment has an associated *purpose*, which we call the Discourse Segment Purpose (DSP) and represent at each node of the tree. Each DSP is a communicative goal of the speaker. In a successful discourse, the contents of each segment achieve its DSP. Each segment can thus be seen as overall communicative purpose of the discourse.

3. Coherence: A discourse is only communicatively successful if it is mutually coherent, i.e., if the speaker's and hearer's beliefs agree about how each segment relates to its neighbors (and thus to the whole). Coherence is enforced by the constraints of intersegment discourse structure relations.

4. Discourse Segment: A discourse segment S is represented by a tuple (name, purpose, content), where:

- The name is a unique identifier for the segment.
- The purpose is one or more communicative goals the speaker has with respect to the hearer's mental state (the DSP)
- The content is either:
  - an ordered list of discourse segments, together with one or more intersegment discourse relations that hold between them (either there is a relation between every two adjacent segments in the list, or a relation holds among all the segments in the list simultaneously); or
  - a single discourse segment; or
  - the semantic material to be communicated (usually statable as a single clause in English). This material often takes the form of a set of knowledge base assertions or data base facts.

The *purpose* is the important thing introduced here. These communicative goals are a function of the document being created (not the data being used to create it). There is still the question of *how* to associate a purpose to the data.

In section 5 the paper describes four major aspects of text planning, all

repeatedly found in the text generation literature, which centrally involve discourse structure relations:

1. Text plans - content and format: The operationalized RST relations themselves were quickly found inadequate, especially in their inability to capture communicative intent. Text planners switched to using a new kind of plan, one keyed on intentionality.

2. A collection of relations: Intersegment discourse relations are still however required to structure the discourse. An ongoing effort to collect and taxonomize a core corpus of relations is described.

3. Predefined structures (schemas): In spite of the utility of text plans and discourse relations, predefined structures remain necessary to control the combinatorics of longer texts.

4. Controlling planning by focus shift: Being able to juxtapose clauses coherently did not mean being able to make them flow successfully. Discourse relations and focus shift rules work together to co-constrain the possibilities.

The paper then goes on to discuss these topics.

They note that the text generators based on intentionality ended up using the same style of processing as those based on RST. This seems to be equvalent to just providing a different set of RST relations, perhaps called intentional relations as opposed to rhetorical relations. I am not sure this achieves much.

> To the extent that a difference does exist, however, the dilemma is resolved when one recognizes that the two types of object - intentional plans and discourse relations - perform different functions and hence are both needed simultaneously to govern discourse. To determine what material to include and to provide the overall structure of the discourse, intentional plans are most appropriate; within this framework, it is the function of discourse relations to ensure textual coherence, prevent unintended inferences, govern sentence formation, tense, pronominalization, and focus shift, as described in subsequent sections of this paper.

I think this means we can use intention at the high levels, above paragraph, and rhetorical relations within paragraphs.

They define a text plan as a tuple (*name, effects, constraints, preconditions, decomposition*):

- The *name* is a unique identifier of the segment.

- The *effects* are one or more communicative goals that the plan achieves, if properly executed. Since these goals pertain to the speaker's desire with respect to the hearer's state of knowledge, opinion, goals, and similar structures, they are phrased in terms of the hearer's mental state.

- The *constraints* are facts in the knowledge base or the user model that must hold before the plan may be used.

- The *preconditions* are facts in the knowledge base or user model that should hold for felicitous communication. If they are violated, the hearer may be confused. As mentioned above, the planner in a dialogue situation may be given the ability to ignore the preconditions, trusting the hearer to request help when communication fails; in such cases, the planner should mark the affected preconditions to facilitate repair.

- The *decomposition* is an ordered list of subgoals to be achieved. Each subgoal may be flagged as optional, in which case the planner can ignore it under appropriate conditions, depending on the planner's sophistication: at the minimum, it can simply ignore the subgoal if instructed to produce terse text; being more sophisticated, it may reason about various contributing factors, such as the balance of material within the discourse structure so far or the level of detail of the indicated material). The order of subgoal segments within this list must respect the coherence requirements of discourse structure relations. Subgoals are generally of two types:

    - communicative intentions on portions of knowledge base contents, which can be achieved by other text plans (for example, a PERSUADE may call for a MOTIVATE or a DESCRIBE), and

    - "primitive" Speech Acts on clause-sized knowledge base entities, such as INFORM, ASK, and ORDER, which are achieved by the sentence generator.

The paper then goes on to discuss taxonomies of rhetorical relations. Various researchers have proposed various collections ranging from just two, to an unbounded set.

They propose that the relations can be grouped into three broad categories: *semantic*, *interpersonal*, and *presentational*. The paper's appendix includes a classification of various relations under these headings.

If SASSY is to use rhetorical relations it would seem that the best option would be to include enough to provide a rich suite of cue phrases that can be applied to the generated sentences.

The next subsection is on schemas. The point is made that trying to use text plans above the paragraph level results in far too many options (i.e. a combinatorial explosion). They suggest that schemas are applicable at this level.

Presumably a schema is limited to simple choices based on the available data. For example, providing all the alternatives for a sequence.

The last part of the section discusses focus of attention. They describe a system that when the text plan has multiple alternatives for its *decomposition* it checks them all against the options from a *focus tree*.

The last section describes the ways in which discourse relations can be used.

1. **Casting of syntactic roles**: An important sentence-level planning task is the assignment of material to syntactic classes within a sentence.

2. **Concept aggregation**: Another planning task involving discourse relations is the compacting of text by aggregation.

3. **Text formatting**: Several discourse structure relations achieve their communicative purposes presentationally using text formatting devices such as itemized lists, headings, and footnotes.

These are useful enough to strongly encourage the use of RST in the text planning process.

A set of hueristics is proposed for sentence formation:

1. A Satellite can only be embedded in its Nucleus.

2. Embedding can be realized as an adjective, appositive NP, PP, or relative clause, in this order of preference.

3. Embedding can occur in the leftmost nuclear clause with the same focus value.

4. Satellites in a LIST within an ELABORATION should be embedded, provided there are no, or else more than one, remaining clauses.

5. Coordination occurs only between elements of LIST, SEQUENCE, and CONTRAST relations.

6. The more shared parameters between clauses, the more they should be coordinated.

7. Prefer coordinating NPs over PPs over Vs or VPs.

8. Sentences should contain no more than 3 clauses.

9. Sentences should contain at most one level of embedding.

10. Embedding should occur before coordination and before focus transformations.

Need to define exactly what *embedding* means.

On the topic of text formatting the paper introduces *textual devices*. These can be classified into three broad classes: *Depiction*, *Position*, and *Composition.* Typical uses for these textual devices are:

1. **Depiction**: selection of an appropriate letter string format.

    - Parentheses: text is tangential to the main text.

    - Font switching: text has special importance (new term, of central importance, foreign expression) when the surrounding text is not italicized.

    - Capitalization: text string names (identifies) an entity.

- Quotation marks: text was written by another author, or some non-literal, special meaning is intended.

2. **Position**: Repositioning of text blocks.
   - Inline: non-distinguished normal case.
   - Offset (horizontal repositioning): text was authored by someone else.
   - Separation (vertical repositioning): text addresses a single point (a paragraph) or identifies subsequent text (headings or titles).
   - Offpage: text provides explanatory material (appendix, footnote).
3. **Composition**: imposition of an internal structure on the text.
   - Itemized list: set of (maximally paragraph-length) discourse objects on the same level of specificity with respect to the subject domain, each more than a clause (e.g., this list of textual devices).
   - Enumerated list: set of (maximally paragraph-length) discourse objects on the same level of specificity with respect to the domain, which are ordered along some underlying dimension, such as time, distance, importance.
   - Term definition: pair of texts separated by a colon or other delimiter, in which the first names a discourse object and the second defines or explains it (e.g., this item on term dejnnition).

Some textual devices with structural definitions are:

- Enumeration: As described in the example above, the text structure relation SEQUENCE can generally be formatted as an enumerated list. The enumeration follows the sequence of the relation, which is planned in expression of some underlying semantic ordering of the items involved, for example time and location.
- Itemization: The textual structure that relates a number of items without any underlying order is the RST relation LIST, which can be realized by an itemized list (unless the items are small enough to be placed into a single sentence).
- Appendix, footnote, and parentheses: These are three devices that realize the same textual relation, namely Satellite.
- Section title or heading: This device realizes the textual relation IDENTIFICATION, which links an identifier with the body of material it heads. A section or subsection is appropriate when the IDENTIFICATION is combined with a SEQUENCE chain that governs the overall presentation of the text.

The paper makes a very good case for using text plans. However it does not

provide much illumination on how one maps the data in the knowledge base to the rhetorical relations. Exactly how RDF statements can be related by rhetorical relations is "left as an exercise for the reader" as they say.

6.5 Building Up Rhetorical Structure Trees

Daniel Marcu, 1996

Abstract: I use the distinction between the nuclei and the satellites that pertain to discourse relations to introduce a compositionality criterion for discourse trees. I provide a first-order formalization of rhetorical structure trees and, on its basis, I derive an algorithm that constructs all the valid rhetorical trees that can be associated with a given discourse.

The paper[7] sets out to

> provide a formalization of the structure of RS-trees and show how one can use it to find answers to the questions:
>
> Despite its popularity, RST still lacks both a formal specification that would allow one to distinguish between well- and ill-formed rhetorical structure trees (RS-trees) and algorithms that would enable one to determine all the possible rhetorical analyses of a given discourse.

The paper sets out to use nuclearity as its basis:

> Despite the lack of a formal specification of the conditions that must hold in order to join two adjacent text spans, I believe that RST contains an implicit specification
>
> During the development of RST, these researchers noticed that which is expressed by the nucleus of a rhetorical relation is more essential to the writer's purpose than the satellite; and that the satellite of a rhetorical relation is incomprehensible independent of the nucleus, but not vice-versa.
>
> A careful analysis of the RS-trees that Mann, Thompson, and many others built shows that whenever two large text spans are connected through a rhetorical relation, that rhetorical relation holds also between the most important parts of the constituent spans.
>
> I propose that this observation can constitute the foundation for a formal treatment of compositionality in RST. More specifically, I will formalize the idea that two adjacent spans can be joined in a larger span by a given rhetorical relation if and only if that relation holds also between the most salient units of those spans.

A modification is introduced into the RS tree - each node has a reference to the most salient node in the subtree that it is the root of. This is the leaf that is the nucleus in each relation below the node.

A set of constraints are proposed:

- For every span [l, h], the set of objects over which predicate S ranges is the set {NUCLEUS,SATELLITE,NONE).
- The status of any text span is unique.

- For every span [1, h], the set of objects over which predicate T ranges is the set of rhetorical relations that are relevant to that span.
- At most one rhetorical relation can connect two adjacent text spans.
- For every span [I, h], the set of objects over which predicate P ranges is the set of units that make up that span.
- Text spans do not overlap.
- A text span with status NONE does not participate in the tree at all.
- There exists a text span, the root, that spans over the entire text.
- The status, type, and promotion set that are associated with a text span reflect the structural and nuclearity constraints.

The paper then goes on to describe an algorithm for generating the valid RS trees. This simply applies the above constraints to a constraint satisfaction solving package, so it is not very enlightening.

From a text generation perspective the paper is not particularly helpful. It assumes the relations are provided and more importantly that the order of the text units is already known. However, it might be useful to apply these constriants to any RS trees that we build.

6.6 Generating Tailored Textual Summaries from Ontologies

Kalina Bontcheva, 2005

Abstract. This paper presents the ONTOSUM system which uses Natural Language Generation (NLG) techniques to produce textual summaries from Semantic Web ontologies. The main contribution of this work is in showing how existing NLG tools can be adapted to Semantic Web ontologies, in a way which minimises the customisation effort while offering more diverse output than template-based ontology verbalisers.

A novel dimension of this work is the focus on tailoring the summary formatting and length according to a device profile (e.g., mobile phone, Web browser). Another innovative idea is the use of ontology mapping for summary generation from different ontologies.

The paper[1] describes work done to create a natural language description of ontologies or knowledge bases.

There are several advantages to using NLG rather than using fixed templates where the query results are filled in:

- NLG can use different sentence structures depending on the number of query results, e.g., conjunction vs itemised list.
- depending on the user's profile of their interests, NLG can include different types of information – affiliations, email addresses, publication lists, indications on collaborations (derived from project information).
- given this variety of what information from the ontology can be included and how it can be presented, depending on its type and amount, writing templates will be unfeasible because there will be too many combinations to be covered.

The processing pipeline is described thus:

Summary generation starts off by being given a set of statements (i.e., triples), in the form of RDF/OWL. Since there is some repetition, these triples are first pre-processed to remove already said facts. In addition to triples that have the same property and arguments, the system also removes triples involving inverse properties with the same arguments, as those of an already verbalised one.

Next is the summary structuring module, which orders the input statements in a coherent summary. This is done using discourse patterns, which are applied recursively and capitalise on the property hierarchy . This module also performs semantic aggregation, i.e., it joins together statements with the same property name and domain, so they are expressed within one sentence.

> Finally, the generator transforms statements from the ontology into conceptual graphs which are then verbalised by the HYLITE+ surface realiser. The output is a textual summary.

The first two processes are of interest for the SASSY microplanner.

The paper makes the point that the text structuring and the lexicon are domain dependent and will need to be set up for each project:

> Typically the text structuring component is domain-dependent, because every domain or application tends to have different conventions for what constitutes a coherent text. Another example domain-dependent module is the lexicon which maps concepts to their lexical items and grammatical information. Therefore, when an NLG system is adapted to a new domain or application, these components need to be modified.

The ONTOSUM system uses label properties in the ontology and hueristics to construct its lexicon automatically. SASSY should be similar.

Discourse structuring uses the schema approach introduced by McKeown[3].

Only the top level schema is described:

> The top-level schema for describing instances from the ontology is:

```
Describe-Instance ->
Describe-Attributes,
Describe-Part-Wholes,
Describe-Active-Actions,
Describe-Passive-Actions
```

> where Describe-Attributes , etc. are recursive calls to other schemas. For example, the Describe-Attributes schema collects recursively all properties that are sub-properties of the attribute-property and involve the given instance:

```
Describe-Attributes ->
[attribute(Instance, Attribute)],
Describe-Attributes *
```

> The schemas are independent of the concrete domain and rely only on a core set of 4 basic properties – *active-action , passive-action , attribute , and part-whole*. When a new ontology is connected to ONTOSUM, properties can be defined as a sub-property of one of these 4 generic ones and then ONTOSUM will be able to verbalise them without any modifications to the discourse schemas.

Aggregation is then performed:

> Once the information from the ontology is structured using the schemas, aggregation is performed to join similar RDF triples. This process joins adjacent triples that have the same first argument and have the same property name or if they are sub-properties of attribute or part-whole properties.

This paper is interesting as it describes work that is recent enough to involve RDF (and OWL). The fact that it relates the semantic RDF predicates to the rhetorical relations gives some hope that SASSY can use a similar technique.

I am not sure about using just four rhetorical relations. A richer set would enable the use of cue phrases in the output text. I think SASSY might be improved if it used the *text plans* described by Hovy[6].

I am also a bit surprised that *focus of attention* did not feature in the solution. This seems like an easy thing to do when working with RDF.

6.7 Natural Language Generation

Eduard H. Hovy, 1991

> **Abstract**: This document reports on research investigating the automatic planning and generation of multisentence text and multimodal presentations by computer. This work was performed from mid-1989 to late 1991 at the Information Services Institute of the University of Southern California (L'SC/ISI). The research consisted, of three stages: development of the basic planning algorithms, collection and synthesis of subsidiary information required for text and multimedia presentations, and design and construction of a new type of communication planner architecture.

This quote should appear in the final program:

> Every day, we effortlessly produce thousands of words of connected discourse from complicated and ill-understood internal knowledge for complicated and ill-understood reasons.

This paper[8] is an early version of [6] which consists of the first part of this one. Hence this review will only cover the second part.

The paper describes a text planning system based on what was learnt from previous attempts. It starts with the knowledge resources required:

> the major knowledge resources that we have so far identified, namely: *text types, communicative goals, schemas, discourse structure relations*, and, finally, a resource to handle *theme development* and *focus shift*.

> In some cases, the knowledge resources actually represent the order of some planning operations. Such resources were implemented as systemic networks; they are the discourse relations and theme patterns. In other cases, the knowledge resources provide information which the planner uses to make decisions. Such resources were implemented as property-inheritance networks; they are the text types, communicative goals, and schemas. Both types of representation are declarative, enabling one to capture inherent commonalities within the resource, and promote notational clarity and simplicity of processing.

Text Types. These are classifications of the type of document being created.

I was going to insert a text type hierarchy here, but the diagram in the paper is indecipherable and web searching was not fruitful. I think SASSY can build its own set of text types by collecting the specifications for each document type that we set up.

> Each text type in this hierarchy has associated with it the constraints it imposes on other resources such as which communicative goals it entails, which discourse relations it favors, any appropriate grammatical constraints, etc. As a result, once a type has been established for the text to be generated, the selection of other parameters used during the generation process can be constrained appropriately.

Communicative Goals.

As have been used in many generation systems, communicative goals describe the discourse purpose(s) of the speaker. The planner contains a rudimentary taxonomization of communicative goals, starting at the topmost level with some very general goals, such as INFORM, DESCRIBE, REQUEST, and ORDER, which are eventually refined into specific goals to describe (or relate, etc.) specific types of information for specific contexts

Schemas.

In many circumstances, texts exhibit a stereo-typical structure. In text planning systems, such structure is usually represented in schemas which specify the topics of discussion that appear in the text as well as their ordering.

In SASSY we will probably only use schemas for the discourse structuring module.

Discourse Structure Relations

These relations must be used in a generation system in order to guide the selection and organization of the information to be included when other structuring guidance is lacking, such as when a schema stage calls for more material than can fit into a single clause. The necessity and use of discourse structure relations in text planners to ensure coherence has been amply discussed.

The relations were divided into three major portions, corresponding to the three major functions of language (semantic/ideational, interpersonal, and presentational/textual);

When organizing material, the planner is free in the general case to establish several discourse relations (typically, one for each of the major functions) between the existing discourse structure and the new piece of material; as shown in the networks, the selection of ideational, interpersonal, and textual relations is not exclusive.

Theme Development Information. This is something that doesn't seem to have been mentioned previously.

Though the study of theme has been traditionally been restricted to the sentence level, it also plays a role at the the clause-complex and even discourse levels. This should be taken into consideration by a text generation system. Given a text to be generated, the system must establish how theme development may proceed and how themes are to be marked in each clause. The following three concerns arise:

- the type of theme to select: following Halliday[9] , there can be three different and simultaneous themes in each clause: the ideational (or topical; expressing processes, participants, or circumstances), the interpersonal (expressing modal meanings such as probability, usuality, or opinion), and the textual (such as continuatives - "yes,", "well,", "oh,", or conjunctions). The first type is semantically required.

- the theme progression pattern involved: the new theme can be the same as the theme of the previous clause; it may be part of the rheme of the previous clause; or it may be an element of what is called the "hypertheme" or general discourse segment topic. Note also the similarity to the focus shift rules of Sidner and McKeown.

- the linguistic degree of markedness of the theme: realization depends on the type of clause.

The motivations behind each choice follow pragmatic principles of information processing, including:

- *the Topic-Comment constraint*, also known as the *Graded Informativeness requirement*: a message is maximally effective if information which is presumed or given in the context is presented before information which is new;

- *the Processibility principle*: a text should be constructed so that it is easy to process in real time, by placing the focus tone group at the end of the clause (the maxim of end-focus) and the "heavy" constituents in final position (the maxim of end-weight);

- *discourse relation requirements*: some discourse relations have a canonical (unmarked) order of surface realization.

There are some concepts here which need further investigation, such as *theme*, *rheme* and *focus tone group*. We might just use focus of attention in the first instance and then try *theme* and *focus tone* in a subsequent iteration of SASSY.

The paper next moves on to describing the planning process.

> Planning with the networks proceeds analogously to the generation of single sentences with Penman: in both cases, the traversal mechanism proceeds through the network, causing traversal choices to be made at nodes (systems, and building a tree-like structure as a result.
>
> Associated with each node in the networks is an inquiry function which queries the environment in order to determine which branch to follow, and a set of realization operators that instruct the planner what to do next.
>
> The planning operation is very simple. After an initial setup phase, the system simply executes a basic planning cycle over and over again until planning is complete. In the setup phase, the user activates the planner with a communicative goal, which causes the selection of a desired text type, and is then posted on the goal stack and simultaneously on the Discourse Structure Tree. Then the basic planning cycle begins. Essentially, this cycle proceeds as follows: First, the planner checks whether there is a realization on the agenda. If so, it performs the realization by applying its action to its parameters. If there are no realizations left, the planner checks whether there is a discourse goal on the goal-stack. If there is, the planner finds the realizations associated with the goal and loads them onto the agenda; if no discourse goals remain, the planning is done.
>
> Each realization is an instruction to be performed. At present, the system

uses the following realizations:

1. (ACTIVATE-SCHEMA schema-name): Find the schema and load its realizations onto the agenda.

2. (ADD-TO-D-STRUC goal concept parentpos): Add the given communicative goal into the discourse structure tree at the given position.

3. (CHANGE-HYPERTHEME -chainofroles-): Change the topic under discussion to the filler of the given chain of roles, starting from the current topic.

4. (HIGHLIGHT-COHN-GOALS -goals-): Highlight the given goals so that only they will be considered for future planning.

5. (HIGHLIGHT-RELATION -relations-): Start traversal of the discourse relations network(s) at the given relations, using the current topic of discussion.

6. (BLOCK-RELATION -relations-): Mark the given discourse structure relations so that they cannot be traversed for the remainder of the current sentence.

7. (PREFER-THEME conceptrole): Add instructions for the realization component that the given role of the topic under discussion should be thematized in the clause.

8. (SET-MACROTHEME concept): Change the overall topic of discussion.

9. (SET-UP-DISCOURSE-GOAL goal): Activate the given goal: load it onto the goal stack and into the discourse structure tree at the current growth point and add its realizations to the agenda.

10. (TRAV-ONE-NETWORK-NODE node-name): Locate the given node in the knowledge resource networks, apply its inquiry function, record the result (the inquiry choice), and load the realizations associated with the result onto the agenda.

The remainder of the paper works through an example, and then digresses into multi-media communications which is outside of our scope.

6.8 The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts

Daniel Marcu, 1997

Abstract: This thesis is an inquiry into the nature of the high-level, rhetorical structure of unrestricted natural language texts, computational means to enable its derivation, and two applications (in automatic summarization and natural language generation) that follow from the ability to build such structures automatically.

The thesis proposes a rst-order formalization of the high-level, rhetorical structure of text. The formalization assumes that text can be sequenced into elementary units; that discourse relations hold between textual units of various sizes; that some textual units are more important to the writer's purpose than others; and that trees are a good approximation of the abstract structure of text. The formalization also introduces a linguistically motivated compositionality criterion, which is shown to hold for the text structures that are valid.

The thesis proposes, analyzes theoretically, and compares empirically four algorithms for determining the valid text structures of a sequence of units among which some rhetorical relations hold. Two algorithms apply model-theoretic techniques; the other two apply proof-theoretic techniques.

The formalization and the algorithms mentioned so far correspond to the theoretical facet of the thesis. An exploratory corpus analysis of cue phrases provides the means for applying the formalization to unrestricted natural language texts. A set of empirically motivated algorithms were designed in order to determine the elementary textual units of a text, to hypothesize rhetorical relations that hold among these units, and eventually, to derive the discourse structure of that text. The process that finds the discourse structure of unrestricted natural language texts is called rhetorical parsing.

The thesis explores two possible applications of the text theory that it proposes. The rst application concerns a discourse-based summarization system, which is shown to significantly outperform both a baseline algorithm and a commercial system. An empirical psycholinguistic experiment not only provides an objective evaluation of the summarization system, but also con rms the adequacy of using the text theory proposed here in order to determine the most important units in a text. The second application concerns a set of text planning algorithms that can be used by natural language generation systems in order to construct text plans in the cases in which the high-level communicative goal is to map an entire knowledge pool into text.

The paper[10] attempts to formalise the rhetorical structure of text sufficiently for it to be computationally tractable. For example there is the strong composition criterion:

> The strong compositionality criterion ... stipulates that if a relation holds between two textual spans of the tree structure of a text, that relation also holds between the most important units of the constituent spans.

Chapter 2 of the paper starts by describing the essential features of text structures:

> the elementary textual units are non-overlapping spans of text; that there exist rhetorical, coherence, and cohesive relations between textual units of various sizes; that some textual units play a more important role in text that others; and that the abstract structure of most texts is a tree-like structure.

Section 2.5 includes a very nice description of how RST works. An important distinction is made for the relations: they are either *paratactic* (having multiple nuclei) or *hypotactic* (having a nucleus and a satellite).

It then goes on to develop a formalisation:

> **Definition 2.2. An extended formulation of the problem of text structure derivation**: *Given a sequence of textual units* $U = u_1, u_2, ..., u_n$ *and a set RR of simple and extended rhetorical relations that hold among these units and among contiguous textual spans that are defined over U, find all valid text structures of the linear sequence U.*

A set of constraints are proposed. These were the same as listed in the review of [7].

It should be emphasised that these constraints refer to text that has a predefined order for the text spans. This is what we are trying to determine in a text generation system. This may still be useful if we have some means of generating the order of the text spans for all, or most, of the text and would like some way of determining which is the better.

The next section looks at formalizing the relationship between text trees and intentions. It develops a new set of formalisations (which need to be further reviewed) and concludes with

> In such a case, the axiomatization provides the means for drawing intentional inferences on the basis of the discourse structure.

This implies that it might be possible to classify various alternative constructs of the text by their inferred intentions. This might be a useful approach if the discourse planner was providing an intentional parameter for the microplanner.

Chapter four looks at automatically deriving the relational tree structure:

> discuss the lexicogrammatical constructs that can be used to determine the elementary units of a text and to hypothesize rhetorical relations among them. These constructs include grammatical morphemes, tense and aspect, certain lexical and syntactic structures, certain patterns of pronominalization and anaphoric usages, cohesive devices, and cue phrases.

While the chapter examines how to use these constructs to identify relations, they will also be applicable when we use the relations to construct the specification for the NLG.

Most of chapter four is about using cue phrases to automatically identify text spans and rhetorical relations.

Chapter five discusses the rhetorical parsing of unrestricted natural language texts. While not directly related to text generation it could prove to be a useful resource when constructing the rules used to turn the raw clauses into input for the NLG.

Chapter six builds on the parsing by creating a summarisation algorithm.

Chapter seven is about text generation and has some useful information for our project. It starts by making the observation that traditional top down planning approaches are not good for reporting systems:

> Unfortunately, this strength is also a major weakness, because top-down and schema-based approaches are inadequate when the high-level communicative goal boils down to "*tell everything that is in this knowledge base*" or "*tell everything that is in this chosen subset*". The reason for this inadequacy is that these approaches cannot ensure that all the knowledge that makes up a knowledge pool will be eventually mapped into the leaves of the resulting text plan; after building a partial text plan, which encodes a certain amount of the information found in the initial knowledge pool, it is highly likely that the information that is still unrealized will satisfy none of the active communicative goals.

The issue of determining the relations that hold between pairs of semantic units is touched upon without providing much guidance. However, other researches have shown that we can just use the RDF predicates, or some mapping thereof.

The basis of the bottom up approach is:

> the bottom-up approach to text planning assumes that global coherence can be achieved by satisfying the local constraints on ordering and clustering and by ensuring that the discourse tree that is eventually built is well-formed.

The author made a study of a corpus of text and derived a set of statistics for each rhetorical relation of the strengths of the local constraints that characterize coherent texts. The data is included in Appendix E of the the thesis. The data items are:

> the strength of the preference for the nucleus to precede the satellite, s_o; the normalized average number of sentences that separate the nucleus and satellite, avg_s ; the average number of clause-like units that separate the nucleus and satellite, avg_c ; the strength of the clustering preference, s_c, which reflects the inclination of rhetorically related units to be realized as adjacent clauses.

The formulae for these statistical values are presented in the thesis.

An algorithm is presented for the generation of the RST. I have implemented this algorithm in C++ and found that it is quite highly dependent on the number of sematic units, with 10 being a practical upper limit for useful run times. Notes are made on how to convert it to a greedy form for better performance, but I have found that this breaks the algorithm. I have

implemented a partial greedy version which keeps more intemediate trees and which works as well as the non-greedy version in a fraction of the time.

The thesis concludes with the following remark concerning text generation:

> A more serious criticism concerns the computational properties of the bottom-up text planning algorithms. It is true that the bottom-up text planning algorithms that I have proposed are only exponential, and not undecidable, as the ubiquitous top-down planning algorithms are, but still if the bottom-up algorithms are to be applied for large scale problems, better solutions will have to be identified.

Note: While the bottom up algorithm creates an RST it is not clear that it is necessarily the correct tree. Just because a sequence of sentences is coherent does not mean that it is conveying the intended meaning.

6.9 Natural Language Generation: An Overview

Paul Semaan, 2011

Abstract: In this paper, we are discussing the basic concepts and fundamentals of Natural Language Generation, a field in Natural Language Engineering that deals with the conversion of non-linguistic data into natural information. We will start our investigation by introducing the NLG system and its different types. We will also pin point the major differences between NLG and NLU also known as Natural Language Understanding. Afterwards, we will shed the light on the architecture of a basic NLG system, its advantages and disadvantages. Later, we will examine the different applications of NLG, showing a case study that illustrates how an NLG system operates from an algorithmic point of view. Finally, we will review some of the existing NLG systems together with their features, taken from the real world.

This document is a brief overview of NLG. It would be a good introduction for anyone starting out in this field.

6.10 Natural Language Generation in the Context of the Semantic Web

Nadjet Bouayad-Agha, Gerard Casamayor and Leo Wanner, 2012

Abstract: Natural Language Generation (NLG) is concerned with transforming given content input into a natural language output, given some communicative goal. Although this input can take various forms and representations, it is the semantic/conceptual representations that have always been considered as the “natural” starting ground for NLG. Therefore, it is natural that the Semantic Web (SW), with its machine-processable representation of information with explicitly defined semantics, has attracted the interest of NLG practitioners from early on. We attempt to provide an overview of the main paradigms of NLG from SW data, emphasizing how the Semantic Web provides opportunities for the NLG community to improve their state-of-the-art approaches whilst bringing about challenges that need to be addressed before we can speak of a real symbiosis between NLG and the Semantic Web.

The paper[11] starts off with an overview of natural language generation. It lists the main tasks for a full NLG system:

1. *content selection* that determines which parts of the content received as input are to be verbalized according to the context;

2. *discourse planning* that organizes the content so that it is rendered as a coherent text;

3. *lexicalization* that maps conceptual (or language-independent semantic) configurations onto language-specific senses or words;

4. *aggregation* that merges partially overlapping content and linguistic structures to avoid repetition and to improve the fluency of the output; see, e.g., the aggregation of the syntactic representations of the two sentences "*The wind will be weak. The rain will be light*" to produce the more fluent text "*The wind will be weak and the rain light*" ;

5. *generation of referring expressions*, i.e., generation of anaphora and generation of references to entities supposedly already present in the reader's world model; and

6. *linguistic realization* that deals with mapping the discourse or sentence specifications obtained from the preceding tasks onto a syntactically, morphologically and orthographically correct text.

In the SASSY model content selection and high level discourse planning will be one module that will be driven by user input. The linguistic realisation will be handled by our NLG module. The remainder is handled by this microplanner module.

The paper provides an extensive survey of the NLG field and how it might be able to leverage the semantic web. It has an extensive bibliography that might be a useful resource.

It does not provide much that is directly helpful for the project.

6.11 Discourse Structures for Text Generation

William C. Mann, 1984

Abstract: Text generation programs need to be designed around a theory of text organization. This paper introduces Rhetorical Structure Theory, a theory of text structure in which each region of text has a central nuclear part and a number of satellites related to it.

A natural text is analyzed as an example, the mechanisms of the theory are identified, and their formalization is discussed. In a comparison, Rhetorical Structure Theory is found to be more comprehensive and more informative about text function than the text organization parts of previous text generation systems.

This paper[12], despite its title, only provides a worked example of RST analysis and goes no further into text generation than saying that RST would be a good idea.

6.12 Approaches to the Planning of Coherent Text

Eduard H. Hovy, 1989

> **Abstract**: This paper discusses the planning of multisentential text by computer. In order to construct coherent paragraphs, we have been using relations from Rhetorical Structure Theory (RST) operationalized, as plans. The paper first describes, in some detail, the current method of planning a paragraph using operationalized RST relation/plans. It then makes two points that illustrate why RST relation/plans are the ideal tool for planning paragraphs. First, these relation/plans can be shown to combine the best features of paragraph-sized schemas and clause-pized planning rules under a top-down planning regime in a way which affords much flexibility to the user. Second, RST relation/plans can support both standard top-down planning and open-ended conversation-like behavior; a small difference in treatment gives rise to either paradigm.

Section two of the paper[13] provides a worked example of using RST to plan a piece of text. The relations are "*operationalised"*, which appears to mean that they have code attached which is able to select data from the knowledge base that conforms to the requirements of the relation. (See the constriants listed for each relation in the Appendix).

Each relation also includes a set of relations that can be "*growth points*". These form new goals for the planning process.

The planning process proposed is:

> the planning cycle is the following: a new growth-point-turned-goal is taken from the agenda; zero or more relations are found to fulfill it; these relations (if any) furnish requirements for nucleus and satellite fillers; if any input units match, the relations are instantiated with the units and added to the tree. Unfulfilled growth points are added to the agenda of goals. When more than one relation/plan can be added, new trees are formed, identical except for the new relations, and the structurer proceeds to plan out all alternatives.

Note that the planning process can finish without using all of the input units. This is not acceptable for a reporting program. However it might be possible to use this procedure as part of the discourse planning stage (and perhaps as a preliminary stage for the microplanner).

Another problem with this approach is that growth points require domain knowledge to construct:

> By developing the notion of growth points, we have managed to achieve a suitable melding. Rather than allowing all possible RST relations to enter at each cycle of the planning process, we allow growth of only those relations whose goals appear in the growth points fields, and we build in the growth points fields only those goals that support the texts commonly found in the domain of discourse.

The implication is that the language generation has to be developed by a domain expert. This is a considerable burden to place on the users of SASSY.

It should also be noted that the generated text is a single paragraph with a well

defined subject. This is OK for our RDF data. How this would scale to an entire document is not clear. RDF data is only connected on a short scale - trying to discover relations between unconnected statements would be difficult. Perhaps the RDF models need to be hierarchical?

The paper sets out the desirable characteristics of their approach. These mostly derive from the declarative way in which the text plans and growth points are defined.

Section four introduces a modification to the processing model. The relations in the growth points are changed from injunctions to suggestions. This results in a more open ended and flexible planning regeme. This is more applicable to conversational interactions rather than report generation.

6.13 Problems in the Application of Rhetorical Structure Theory to Text Generation.

Nick Nicholas, 1994

> The application of RST to text generation has highlighted a number of problems with the theory; these problems had not been as disruptive while RST was limited to the descriptive domain. I believe these problems affect not only the analytical use of RST, but its computational use as well.

The paper[14] describes three problems. The first is the scope of RST trees:

> As a result, the RST trees drawn by text generation researchers consider nominalisations, relative clauses, and even adjectives as satellite text spans, to be linked to nuclei within the main clause. Classical RST would consider these text spans as embedded within their matrix clauses, and thus not subject to rhetorical analysis.

This raises the question of where aggregation should occur in the process. If some RDF statements can be aggregated prior to trying to construct the RST tree it may take some of the load off that process.

The second problem is that the set of relations is arbitrary. This can lead to an ever increasing set until they become indistinguishable from the propositions in the knowledge base.

Thirdly, the open ended set of relations and the tendency for analysts to fine tune the set according to the texts they are working with results in

> If each analyst is allowed to choose their own set of rhetorical relations, RST analyses become no longer reproducible

In section two the paper starts by dividing the RST relations in *presentational* and *informational*. The point is made that this does not affect text generation, but I think that the top level of text plans would be driven by the document type (i.e. are presentational) while the lower level ones would be more driven by the knowledge base.

Section three goes into deep arguments about taxonomies and such. The discussions may be quite useful for determining what cue phrases to use in the text generator.

The section also highlights the different types of communications that are in common use. SASSY will not be trying to make convincing arguments or providing directions on how things should be done. Its text generation will be all about what exists.

Section four is a review of the Scott & de Souza programme[15]

In interesting observation:

> But establishing coherence is quite different work from the other half of the problem: making the coherent text rhetorically unambiguous.

This might indicate a problem with a system that automatically creates the RST tree from a collection of semantic units where the result is coherent but perhaps conveys the wrong meaning.

The section discusses, in depth, the use of signalling of relations by the use of cue phrases and connectives. This should be reviewed when dsigning this aspect of the text generator.

6.14 Text Generation Starting From an Ontology

Dragoş Alexandru Cojocaru, Ştefan Trăuşan-Matu, 2015

> **Abstract**: The subject of this paper is the development of an application which generates natural language text, starting from an OWL ontology. The Natural Language Generation, in the context of Semantic Web, represents a relatively new field of research, but due to the capabilities of the ontologies (central element of the Semantic Web) of being dynamically modified and completed with new information, the theme of the application is of great importance. The project employs Rhetorical Structure Theory to structure hierarchically the ontological content, resulting in a human-like discourse structure. Since the Semantic Web is continuously adding machine-readable content, the user can take advantage of this impressive database of knowledge transformed into coherent texts for human with the aid of our application.

The paper describes an (unnamed) application that reports on an OWL ontology for wine.

The process begins from a selected class or individual. From there it selects the related classes (but not related individuals). It has a data processing phase where the ontology is unraveled into a set of RDF statements where all objects are named.

Next it generates sentence plans (perhaps "clause plans") using templates for each property. This is followed by the generation of referring expressions (e.g.

pronouns) and aggregation. Their aggregation rules include the situation where the subject and predicate are the same. However they can also aggregate with differing predicates, though the rules for doing so are not specified.

The say they use RST but their text plan seems very rigid, perhaps better described as a schema.

6.15 Getting the Message Across in RST-based Text Generation

Donia Scott, Clarisse de Souza, 1990.

> Note: Paper only available as an image which is really annoying.

> This chapter examines the problem of generating texts that achieve their communicative goals in an effective way. We discuss an approach to producing effective text that is geared towards ensuring that the rhetorical aspects of a message are not only preserved but enhanced in the text. This approach is strongly influenced by research in psycholinguistics and the psychology of memory, and is based on a view of stylistics as a matter having rather more to do with cognition than aesthetics.

The paper puts forward three basic requirements for the generated text:

> First they must be sensitive to the communicative context in which they are set. Second, the chosen expression of the message must be a valid and unambiguous rendition of its rhetorical structure. Third, the chosen expression must be the most easily processable member of the set of all valid and unambiguous expressions of the message.

For the first requirement they put forward the hypothesis:

> Readers are unlikely to retrieve the rhetorical structure of a message unless it is stated explicitly.

To satisfy this they propose the following heuristic #1:

> Always generate accurate and unambiguous textual markers of the rhetorical relations that hold between the propositions of the message.

This leads to a requirement on text generators that they include information about the appropriate textual markers for each relation. The information should include not only the actual marker phrase but whether it can be used inter- or intra-sententially, does it apply to the nucleus or satellite, and how it might depend on the ordering of the nucleus and satellite.

The paper then goes on to discuss the strength of markers with *and* being the weakest. This may not be as important in a report generator since we can use

fixed markers for each relation. This may make the text a bit more boring to read, but it will not introduce any ambiguity in the mind of someone try make a technical interpretation of the text. This is important for an application aimed at assisiting software development.

The next topic centres on keeping the text easily comprehensible by avoiding long distances between the dependent clauses. Their first hypothesis is:

> The greater the amount of intervening text between the propositions of a relation, the more difficult it will be to reconstruct its message.

The resulting heuristic #2 is:

> Keep the propositions of a rhetorical relation together in the text.

The second hypothesis is:

> Rhetorical relations that are expressed within a single sentence are more easily understood than those expressed in more than one sentence.

I am a bit unconvinced by their corresponding heuristic #3:

> Make a single sentence out of every rhetorical relation.

Other researchers put a limit of just three propositions in a sentence, though this does not usually include elaborations that are aggregated as adjectives.

The authors make the following observation:

> The hueristic proposes that parts of a rhetorical relation should not be realised as individual sentences; neither should they be combined as sentences with parts of other rhetorical relations. Rather, they should all together form a sentence.

If we assume, as has been made clear by many other papers, that the rhetorical relations form a binary tree covering the entire text then it is quite difficult to understand this. The paper goes on to make a similar statement without offering any useful solution to determining where to put sentence boundaries.

Section three then continues discussing ways of combining clauses into complex sentences. However it only covers *embedding* and *paratactic coordination* and leaves *hypotactic coordination* to another paper.

They have proposed the following heuristic #4 for embedding:

> Embedding can only be applied the the ELABORATION relation.

And heuristic #5:

> When embedding, the nucleus of the relation must form the matrix of the sentence and the satellite the embedded clause.

If the nucleus of the ELABORATION relation is complex then heuristic #6 should be applied:

> When embedding, the matrix proposition must be the earliest occuring in the immediate nucleus of the to-be-embedded proposition.

Heuristic #7:

> Propositions of a LIST relationship should not be embedded if doing so would make the number of remaining propositions in the relation equal to one.

The idea with #7 is to prevent dangling sentences.

The paper then makes the point that the choice of syntactic realisation for the embedded satellite is important.

> Embedded clauses can be realised as nominals, adjectivals or adverbials. Although the choice between these realisation classes will be determined by strictly semantic aspects of the propositions there is a still a choice to be made regarding the most appropriate syntactic form within the chosen class. Adjectivals can be expressed as an adjective, a relative clause, or a prepositional phrase; adverbials as an adverb or prepositional phrase; and nominals as a noun or an appositive phrase.

This is covered by heuristic #8:

> Syntactically simple expressions of embedding are to be preferred over more complex ones.

The final embedding heuristic #9:

> Self-embedding is only allowed in cases where the proposition that is the deeper of the two is expressed as an adjective or adverb.

This has the nice effect of precluding sentences like *The dog that the cat that the rat saw chased died.*

The remainder of the paper considers paratactic coordination (where the elements have equal status).

The criteria for determining which relations it can be applied to is given in heuristic #10:

> Paratactic Coordination can only be applied to multi-nuclear relations.

These relations are SEQUENCE, LIST, CONTRAST and ALTERRNATIVE. The conjunctions are specified in heuristic #11:

> The paratactic marker *and* must only be applied to SEQUENCE and LIST, *but* to CONTRAST, and *or* to ALTERNATIVE.

Heuristic #12:

> Propositions of all relations except SEQUENCE can be reordered during paratactic coordination.

SEQUENCE can not be reordered as it has a temporal aspect.

Where the propositions have multiple common elements heuristic #13 applies:

> The greater the number of shared elements between propositions, the more desirable it is to coordinate them.

I am inclined towards going to bullet points or itemised lists if there are more than about four propositions. Coordination can be overdone.

6.16 Sentence Generation And Systemic Grammar: An Introduction

John A. Bateman, 1997.

> **Abstract**. This paper provides a first introduction to some basic issues in sentence generation. It describes how systemic-functional grammars deal with the problems of sentence generation and gives a brief overview of the use of systemic-functional grammars in sentence generation research.

Section two discusses some problems that can happen in text generation. Several examples are provided that consist of propositions in the correct sequence and sentences that are grammatically correct but which are still nonsensical.

Section three looks at some simple ideas for generation where the function of the text restricts the choices available.

> In particular, we can attempt to organize the mapping around the *communicative functions* that the structural realizations achieve. In other words, we try to add information concerning when it is appropriate to use one of the possible ways of expressing a concept rather than another. A straightforward first approximation is shown (5):
>
> •asserting / questioning / ordering:
> e.g., *the window is open, is the window open?, open the window!*
> •positive / negative:
> e.g., *the window is open, the window is not open*
> •not evaluated / evaluated (e.g., modalized):
> e.g., *the window is open, the window might be open*
> •express 1 place relation / express 2 place relation:
> e.g., *the window was open, I opened the window*
> •express 2 place relation with agency / ... without agency:
> e.g., *I opened the window, the window was opened*
> •foreground the agent / background the agent:
> e.g, *I opened the window, the window was opened by me*
> •foreground the affected / do not foreground the affected:
> e.g., *The window I opened, I opened the window.*

Section four, the bulk of the paper, introduces systemic-functional grammar (SFG).

in a functional approach such as SFG, it is the paradigmatic description that is central; here it is the functional aspects of the classifications exemplified in (5) that receive the most attention. This organization has been found to help significantly in the construction of NLG systems and in the formulation of the mapping between meaning and form.

The most important feature of SFG is then its organization around the concept of 'choice'. Within SFG all grammatical variation is captured by abstract choices between minimal grammatical alternatives of the kind illustrated in (5). Moreover, all strata of the linguistic system—i.e., phonology, grammar, semantics—are represented within SFL as paradigmatically organized resources. The central representation adopted for this, which is found in all systemic approaches, is called the system network.

The next subsection describes *system networks*.

A system network is a directed graph with labelled arcs whose nodes are choice points—the 'systems' from which systemic theory takes its name.

The outward directed labelled arcs of each system denote the output features, or *terms*, of that system. Each system has two or more terms, which at the stratum of grammar represent grammatical alternations

The inward directed arcs for each system denote an *entry condition* which determines the paradigmatic context in which the alternation represented by the system is relevant. Entry conditions can consist of disjunctions and conjunctions of output features from other systems; negation is not typically employed and the graph is generally required to be acyclic in computational approaches.

Systems can also share entry conditions, indicating that they are all relevant in the paradigmatic context described by the shared entry condition. Such systems are called *simultaneous*...This captures the fact that functional discrimination may proceed along several independent dimensions in parallel: it is not enough to decide whether a clause is asserting, ordering or questioning, it is also necessary to decide what kind of semantic event is being generated. Following any one path leaves the linguistic unit being described underconstrained with respect to the options presented by other simultaneous paths.

One final kind of organization inherent in the system network is useful both linguistically and computationally for resource construction and maintenance. This involves the central systemic notion of *metafunction* (Halliday 1978). Metafunctions divide a grammar into three semantically motivated areas: one concerned with the expression of propositional content (called ideational in SFL), one concerned with the interaction between speaker and hearer (in terms of asking questions, giving orders, being polite or familiar, expressing opinions, etc.), and one concerned with the textual construction of a stretch of natural language (in terms of anaphors, focusing devices, ellipses, etc.).

The three metafunctional areas of a grammar are interlinked in the normal way—i.e., simply by simultaneity of entry conditions.

Sadly the paper becomes a bit impenetrable after that subsection. As far as I can make out it seems to be describing a system for building actual English sentences, which is something we are delegating to the NLG module.

6.17 Managing Sentence Planning Requirements

Eduard Hovy and Leo Wanner, 1996

Abstract: In this paper we focus on sentence planning (microplanning) as a distinct phase of the generation process, necessary to compute such aspects as sentence delimitation, sentence-internal organization, reference, etc. Treating each phenomenon as a separate planning task, we illustrate the need for a planning paradigm different from the traditional ones in (at least) this phase because of the complex interactions among these (semi-independent) subtasks. We describe our solution: a blackboard-based sentence planner, in which different subtasks are handled by independent modules that incrementally rewrite and flesh out a text structure (a set of partially specified input statement(s)) to form a fully specified list of sentence specifications.

The paper describes the high level design of a sentence planning system.

It starts by listing some of the tasks to be performed:

> **Fine-grained discourse structuring.** Since the precise boundary between text and sentence planning remains an open question, it is likely that some fine-grained, local, discourse structuring tasks remain for the sentence planner. Issues include the ordering of locally related clauses, and the inclusion or exclusion of explicit discourse markers.
>
> **Sentence content delimitation.** The sentence planner's task is to apportion the information that must be said into distinct sentences. Little computational research has addressed this question to date, although for work on the determination of temporal, spatial, and causal nuances of predicates
>
> **Internal sentence organization.** Within a sentence, the sentence planner must allocate the subject, specify the adjuncts (if any), determine the order of preposition phrases, determine the subordination of relative clauses, and so on. These tasks are somewhat interrelated, and are not independent of other high-level sentence planning tasks such as reference and lexical choice. Issues here involve theme and focus , nuclearity (as used in Rhetorical Structure Theory), salience, phrase order, redundancy, and so on.
>
> **Reference choice.** Depending on one's approach, the task of reference can be seen as the central issue in sentence planning (once one has specified how to name each entity and each event, the rest follows) or as a more restricted issue (as endophoric lexical choice that involves just pronominalization and

other NP forms). The relative merits of the range of possible positions remain unclear.

Lexical choice. In its narrowest construal, this task involves selecting from a set of semantically equivalent but potentially pragmatically and syntactically different alternatives for entities that have not been mentioned before in the discourse (exophoric lexical choice). Taken broader, it involves both exophoric and endophoric choice. A considerable amount of computational research has been done on various aspects of lexical choice.

The second section starts with a list of requirements for a Sentence Planner, the first is a functional requirement, the others are more to do with the maintainability of the result:

1. The SP must transform an underconstrained input of deep semantics, as given by the text planner, into a suitably constrained output of shallow semantics, as is often required by the realization component.

2. The SP must modularize sentence planning tasks as far as possible, to facilitate design clarity, development, and maintenance. Since the sentence planning tasks are not single-step operations, since they do not have to be performed in strict sequence, and since the planner's operation is non-deterministic (that is, early choices may undergo subsequent revision), each sentence planning task should be implemented by a separate module.

3. The intermediate steps of the SP should be accessible and easily interpretable to the builders of the SP, to enable cross-module interconnection and debugging.

4. The SP must be extensible, allowing new modules to be introduced as needed.

5. The level of sophistication of the knowledge within a module must not be constrained by the SP architecture, so that the modules might be crude initially but then can incrementally be re ned without impeding throughput. To facilitate this, the rules and knowledge resources employed by the SP modules should be represented as declaratively as possible.

There is a short discussion about the technologies used. It does seem that they are using things that have already been developed. It is annoying that there is no open source version of these items.

The system is constructed of the following modules:

1. A Set of Sentence Planning Task Modules : Discourse Structuring, Aggregation, Phrase Ordering (these latter two are part of the Sentence Structuring task), Exophoric Lexical Choice, and Endophoric Lexical Choice.

2. Knowledge Resources : the lexicon, the semantic model (split into the Upper and Domain Models), the Reader Model, and the lexicogrammatical resources.

3. The Blackboards : the main blackboard (MBB), which contains the latest

message expression(s) together with their derivation history, and the control blackboard, which contains bookkeeping information such as the flags that signal the status (running/idle/etc.) of each module.

4. The Administrator : the top-level process that invokes modules, updates message expressions, and manages parallel alternative expressions, etc.

5. The Tree Transformer : the engine that rewrites (portions of) a message expression as specified by a tree-rewriting rule.

6. The Network Traverser (NetMaster): a process that traverses the system network of each module, handles the inquiry choice functions (typically, these criteria pertain either to the input message or to one of the knowledge resources), and upon reaching tree-rewriting rules, hands them on to the Tree Transformer.

The operation of the system is described thus:

> The planning process starts when the Administrator places a TSL fragment onto the MBB and activates the first module on the Agenda.
>
> Upon activation, a module is handed a (possibly fragmentary) pre-SPL expression from the MBB by the Administrator. After operating upon it, the module replaces the resulting pre-SPL(s) back on the MBB. During the operation, pre-SPLs are transformed to form larger and more complete (more detailed) pre-SPLs. The planning process ends when the MBB contains no more pre-SPL(s) to be completed|i.e., when no module can make any further changes to any (pre-)SPL.
>
> Once a module is activated with a pre-SPL fragment, the NetMaster's task is to determine which tree-rewriting rule(s) to apply. It does so by traversing the module's system network, executing its choice point decision routines (the inquiries), and passing any tree-rewriting rules encountered as realizations to the tree transformation engine. This engine matches the left hand side of each rule to the current pre-SPL expression, and, if successful, unifies all variables and transforms the matched portion of the expression to the rule's right hand side. It returns to the NetMaster the newly constituted pre-SPL expression.

The paper concludes on a rather depressing note:

> Despite previous and current text planning work, we are skeptical that straightforward applications of AI techniques will suffice for planning in NLG in general. The phenomena are too interwoven; the problems are too complex. Although past attempts to factor out various aspects of the generation process and to focus on them individually have achieved some successes (notably in the areas of surface realization, content selection, and paragraph-length text structuring), the more tightly interwoven subtasks of sentence planning remain to be dealt with. Naturally, they can (and should!) be studied in isolation, as in the above-referenced work. But it is important also to study them in concert, in an end-to-end generation system, precisely in order to determine their interdependencies and their interactions.

6.18 Towards A Computational Theory Of Definite Anaphor Comprehension In English Discourse

Candace Sidner, 1979

Abstract: This report investigates the process of focussing as a description and explanation of the comprehension of certain anaphoric expressions in English discourse. The investigation centres on the interpretation of definite anaphors, that is, on the personal pronouns, and noun phrases used with a finite article *the, this* or *that*.

Focussing is formalised as a process in which a speaker centres attention on a particular aspect of the discourse. An algorithmic description specifies what the speaker can focus on and how the speaker may change the focus of the discourse as the discourse unfolds. The algorithm allows for a simple focussing mechaism to be constructed: an element in focus, an ordered collection of alternate focii, and a stack of old focii. The data structure for the element in focus is a representaion which encodes a limited set of associations between it and other elements from the discourse as well as from general knowledge.

This description of focussing allows the following hypothesis of anaphora comprehension to be stated and supported. Definite anaphora are signals which the speaker uses to tell the hearer what element in the discourse is the current discourse focus; at the same time, the element in focus constrains which anaphoric expressions can be used to signal the focus. This hypothesis is supported by five results which are presented in this report:

- a means for distinguishing definite noun phrase used anaphorically from those used non-anaphorically.
- a means for distinguishing pragmatic anaphora from bound variable and inter-sentential anaphora.
- rules which use the focussing mechanism for the interpretation of pragmatic anaphora.
- reduction of the search for inferences which support the interpretation chosen for an anaphor.
- a data structure which represents the element in focus and indicates which items can be associated with the focus and which phrases can be used to mention those items.

This report also establishes other constraints which are needed for the successful comprehension of anaphoric expressions. The focussing mechainism is designed to take advantage of syntactic and semantic information encoded as constraints on the choice of anaphora interpretation. These constraints are due to the work of language researchers; and the focussing mechanism provides a principled means for choosing when to apply the constraints in the comprehension process.

A rather long abstract, and it had to be retyped because this is another of those papers only available as an image. I wish more were fed through OCR processing.

The paper deals with the knowledge acquisition aspect, but we are interested in text generation which is the reverse process. However it appears to have many useful insights which I will attempt to note as ideas for the generator.

Definition: Words which point back to expressions previously used are called *anaphoric expressions*, and the words they point to are the *antecedents*.

The paper makes the point that the antecedent is the object that has focus when the anaphor is mentioned. We can reverse this and apply an anaphore if the object in focus is mentioned again.

Noun phrases may not always refer to actual objects in the real (or imagined) world. To clarify, using RDF notation, ex:Spot might represent an actual dog, while a blank node with an rdf:type of ex:Dog represents some dog, but both are noun phrases "Spot" and "a dog".

An anaphore can be either a pronoun, such as *her,* or use the class name, such as *the dog*. An anaphore has to agree with its antecedent in person, number and gender. A hueristic is that the antecedent is the last noun phrase that passes the person, number and gender test. However this doesn't always work:

> The city councilmen refused the demonstrators a permit because *they* feared violence.

The basic hypothesis of the paper [p19] is that the focus of attention acts as an index function for the referring expressions. We can use the reverse of this and say that if a noun phrase is a repeat of whatever has the focus then it is a candidate for replacing with an anaphora.

The paper makes the claim that focussing provides an explanation for the definite anaphora of *this* and *that*.

On p27 the point is made that some anaphoric expressions do not have antecedents that are directly in the text. I think that in the generator this would be represented by a blank node of some class anway and no action would be required.

The grammatical terms *cleft* and *pseudo-cleft* are introduced. For example:

> A cleft sentence is a sentence that is cleft (split) so as to put the focus on one part of it. The cleft sentence is introduced by it, which is followed by a verb phrase whose main verb is generally be. The focused part comes next, and then the rest of the sentence is introduced by a relative pronoun, relative determiner, or relative adverb. If we take the sentence *Tom felt a sharp pain after lunch*, two possible cleft sentences formed from it are *It was Tom who felt a sharp pain after lunch* and *It was after lunch that Tom felt a sharp pain*.

> Pseudo-cleft sentences (also called *wh*-clefts) are similar in function to cleft sentences, but they are formed with the pronoun *what* (= the thing(s) that/which). The emphasis in a pseudo-cleft sentence is on the phrase after the *what*-clause + be: *What you need is a good sleep.*

An expected focus algorithm is proposed:

Choose an expected focus as:

1. The subject of a sentence if the sentence is an *is-a* or a *there-insertion* sentence.
2. The first element of the default expected focus list computed from the thematic relations of the verb, as follows:

   Order the set of phrases in the sentence using the following preference schema:

   - theme unles the theme is a verb complement in which case the theme from the complement is used;
   - all other thematic positions with the *agent* last;
   - the verb phrase.

6.18.1 Terminology

This paper introduces lots of terms which can overwhelm the reader. This is an attempt to build some sort of glossary.

anaphoric expressions	words that point back to previous words
antecedents	words pointed to by anaphoric expressions
bundle	noun phrase and it interpretation based on syntax and sematics
c-command	A c-commands B if and only if B is either a sister of A or dominated by a sister of A. A pronoun cannot refer to a proper name if it c-commands it.
cataphoric reference	forward reference
cleft sentence	a sentence that is cleft (split) so as to put the focus on one part of it. The cleft sentence is introduced by *it*, which is followed by a verb phrase whose main verb is generally *be*. The focused part comes next, and then the rest of the sentence is introduced by a relative pronoun, relative determiner, or relative adverb.
co-specify	a definite anaphore co-specifies the bundle to which it refers.
definite	the reader can identify the object intended by the writer.
definite anaphor	refers to a specific thing. "The man..."
defnp	abbreviation of definite noun phrase
specific defnp	refer uniquely to one entity (which might exist)
generic defnp	refers to a class
attributive defnp	describes but does not refer

deixis	the use of general words and phrases to refer to a specific time, place, or person in context, e.g., the words tomorrow, there, and they
exophoric reference	reference to object external to the conversation
focus	that which is being discussed
implicit contexts	contexts additional to the one from the linguistic expression/
pragmatic anaphor	Has nonlinguistic context, eg "that" while pointing.
prosodics	features such as intonation, rhythm, pitch and stress
pseudocleft sentence	similar in function to cleft sentences, but they are formed with the pronoun *what*
r-ambiguous	ambiguous in reference
specification	relation between a bundle and some database object
theme	the first part of a sentence. Sets the scene.
u-ambiguous	ambiguous in use
ALFL	Alternative Focus List
CF	Current focus
DEF	Default Expected Focus list
PFL	Potential Focus List

7 Discussion

7.1 Rhetorical Structure Theory

An option for the basis for discourse structuring is Rhetorical Structure Theory as initially proposed by W. Mann, and S. Thompson in 1988[4]. (As per the discussion in "Problems in the Application of Rhetorical Structure Theory to Text Generation", N. Nicholas, 1994[14], I will not distinguish between Volitional and Nonvolitional relations.) See appendices A and B.

There are two ways that a Rhetorical Structure Tree (RST) can be constructed: top down or bottom up. The top down approach starts with a RST relation that expresses the intent provided by the Discourse Context and finds a statement that matches. It then uses "growth points" to expand the tree downwards. Growth points are sets of RST relations that are allowable for the particular nucleus and satellite nodes.

This would be a good scheme to try if there was a definitive set of growth points with defined RST relations. Unfortunately it appears that they are problem specific and need to be set up for different scenarios. This is probably beyond the abilities of someone who just wants to design some software.

The top down algorithm also has the possibility of not including some statements if they don't match any of the specified growth points. This is a serious problem for something that is supposed to create reports on the design of software systems. (The same problem also applies to text generation systems that use schemas.)

The bottom up algorithm avoids the missing data problem. However it requires the RDF statements to be in a linear list and there is no mention of how that order is to be achieved.[7] Trying all N! alternatives is not going to be an option, unless we wait until the Universe freezes over.

7.1.1 RST Relations

The relations used in RST are between clauses. They relate two clauses together and are therefore dependent on both clauses. Since the clauses are derived from RDF statements we would need to develop relations between every pair of statements that might occur for some particular subject or object. (I am assuming there will be a common subject or object, otherwise it is difficult to see how they could be related at all.)

This would be too large a burden to put on the users of SASSY.

However, at least one researcher mapped the predicates to relations and reported satisfactory results. Using this idea I found that it works quite well with RDF data. Adding in that the subject or objects have to also match we can use relations assigned to the RDF model's properties. The user is still required to create a relationship between their domain's properties and the set of

rhetorical relations defined for SASSY. This should not be a significant burden on the user.

7.1.2 RST Trees

The conventional top down approach is driven from a set of rules about which relations can be in the child nodes of a node with a particular relation. These rules then drive the data selection from the knowledge base. The problem here is that the rules are very domain specific and they may not retrieve all the available data.

The bottom up approach as proposed by Marcu is not workable. Even if the order of the clauses is already defined it creates an impossibly large set of alternate trees. Selecting a tree that would support the feeling that the text was planned is something that does not seem to appear in the literature.

A third way would be to insert all the clauses into a tree. Starting with a list of clauses that has been arranged into a focus tree, insert each one into the tree. The focus tree order, combined with the connectedness of RDF, would ensure the tree always had only one root.

The tree would be initialised with a set of rules that described which relations could be children of a given relation. This would be dependent on the communicative goal of the text passage and provided as context by the discourse planner.

For each clause, its verb (ie the RDF predicate) would be used to determine a set of concepts. These would then be used to select the relations that the clause could participate in. The tree would then be examined to find where the clause could be inserted. Some ranking of the relations would be used to resolve alternatives.

7.1.3 Data Driven Text Plans

The bottom up approach to deriving an RST tree from the data is only useful for small data sets (typically no more than 10 clauses). It also has the problem that while the resulting ordering of the clauses is coherent there is no guarantee that the order is the correct order. Changing the order of sentences can profoundly alter the meaning of a paragraph.

The conventional top down approach using growth points has the problem that it might not report unexpected data. If no growth points are applicable the data gets ignored. This is not acceptable for the SASSY application.

When we (as humans) construct a text we start from the information we want to present. From there we devise a text plan and then order the clauses accordingly. The important point is the the text plan is derived from the data set to be presented.

A small experiment confirmed that this works quite well.

It was noticed during the experiment that the sentences only use rhetorical relations derived from their predicates. This leaves the problem of how to include those relations that are derived from the required document format or derived from the reader model.

A possible solution might be to classify the RDF input statements into various classes that are, themselves, related using rhetorical relations. I would expect that these relations can be encoded into the SASSY framework and would be established by the discourse planner.

Each of these classifications would then form a subgraph. The algorithm used in the above experiment would then be applied to each subgraph.

7.2 Graph Theory Approach

The hypothesis is that since we are starting with RDF there are relations defined already. If we can make a few classifications of the RDF predicates then it might be possible to use a little Graph Theory to organise the RDF statements in a coherent manner.

Appendix D shows an experiment where this is tried. While a sample of one is hardly proof, it does appear that with an appropriate set of rules it should be possible to produce text that is not nonsense.

7.2.1 Graph Flattening

The RDF instance text for a paragraph could easily have several dozen statements. If we have used a reasoner so that all the inverse predicates are also included we have a significant number of statements.

The algorithms for creating a Rhetorical Structure Tree start with the leaves of the tree in a fixed order.

> **ONTOSUM**: Next is the summary structuring module, which orders the input statements in a coherent summary. This is done using discourse patterns, which are applied recursively and capitalise on the property hierarchy. This module also performs semantic aggregation, i.e., it joins together statements with the same property name and domain, so they are expressed within one sentence.
>
> What is a *discourse pattern*? When given a set of statements about a given

> concept/instance, discourse schemas are used to impose an order on them, such that the resulting summary is coherent. For the purposes of our system, a coherent summary is a summary where similar statements are grouped together.

> **ONTOSUM**: Data properties frequently don't have a range that provides much semantic information (string, int, etc.). It is therefore necessary to annotate the property with this information.
>
> The disadvantage of the this approach is that it requires the user to create manually these mappings for each problematic datatype property. However, the number of such properties is often quite small and, in our experience, it is feasible to do that in cases when the ontology itself cannot be modified.
>
> However, while the problem with the implicit semantics of the datatype properties has been solved, the resulting summary is no longer so concise. One solution, to be implemented in future work, would be to implement a syntactic aggregation component which merges two sentences when they have the same subject and verb.

The semantic aggregation stage joins all propositions which share the same focused entity and relation, so the resulting more complex propositions can have three or more entities that need to be enumerated in the same sentence.

There is a tension between lists and conjunctions. Too many lists can make the document look messy, but too many coordinates in a conjuction is also not desirable. It is apparent that some decisions by the microplanner may need to be fed back up to the higher level processes, such as the discourse planner.

ONTOSUM: Summary Structuring

Discourse/text schemas, as introduced by [5], are script-like structures which represent discourse patterns. They can be applied recursively to generate coherent multisentential text satisfying a given, high-level communicative goal. Each schema consists of rhetorical predicates (e.g. comparison, constituency) which encode communicative goals and structural relations in the text. Rhetorical predicates are also associated with a semantic function which selects appropriate statements from the ontology. In this way, by selecting and instantiating schemas, a text structuring component can produce coherent texts which satisfy given communicative goals.

In more concrete terms, when given a set of statements about a given concept/instance, discourse schemas are used to impose an order on them, such that the resulting summary is coherent. For the purposes of our system, a coherent summary is a summary where similar statements are grouped together.

The top-level schema for describing instances from the ontology is:

```
Describe-Instance ->
```

```
        Describe-Attributes,
        Describe-Part-Wholes,
        Describe-Active-Actions,
        Describe-Passive-Actions
```

where Describe-Attributes , etc. are recursive calls to other schemas. For example, the Describe-Attributes schema collects recursively all properties that are sub-properties of the attribute-property and involve the given instance:

```
Describe-Attributes ->
        [attribute(Instance, Attribute)],
        Describe-Attributes *
```

The schemas are independent of the concrete domain and rely only on a core set of 4 basic properties – active-action , passive-action , attribute , and part-whole . When a new ontology is connected to ONTOSUM, properties can be defined as a sub-property of one of these 4 generic ones and then ONTOSUM will be able to verbalise them without any modifications to the discourse schemas. However, if more specialised treatment of some properties is required, it is possible to enhance the schema library with new patterns, that apply only to a specific property.

Once the information from the ontology is structured using the schemas, aggregation is performed to join similar RDF triples. This process joins adjacent triples that have the same first argument and have the same property name or if they are sub-properties of attribute or part-whole properties.

8 Lexicon

The lexicon is a database of words. It contains the words themselves, the parts of speech, as well as synonyms and concepts as would be found in a thesaurus.

The NLG makes reference to a lexicon. Currently it uses a simple XML file, but needs something more extensive and complete.

The microplanner needs to convert the concepts in the RDF models into appropriate words. For the core of SASSY we can include a prebuilt lexicon for the software architecture domain. However each project will need to create its own lexicon.

> ONTOSUM: The lexicalisations of concepts and properties in the ontology can be specified by the ontology engineer, be taken to be the same as concept names themselves, or added manually as part of the customisation process. For instance, the AKT ontology provides label statements for some of its concepts and instances, which are found and imported in the lexicon automatically

8.1 Source Data

The knowledge database will have label statements for many of its concepts.

The names given to properties, classes and instances can be parsed to a word or phrase.

Many concepts will include descriptions. These words can be added to the lexicon.

Projects should include a data dictionary which will be another source of domain specific words.

8.2 Construction

The words can be used to get their descriptions, parts of speech, etc from an extract from enwiktionary, the online dictionary available for download.

Additional information can be obtained from Wordnet.

It might be useful to parse a copy of Roget's Thesaurus.

This information can then be loaded into an RDF model hosted on Postgresql.

> ONTOSUM: By default concepts are assumed to be lexicalised as nouns and properties as verbs. This is a rather strong simplification, but given that it is true in many cases, it does save the user the effort of having to specify these manually for the entire ontology. Instead, the user only needs to verify that the automatically assigned part of speech is correct and only change the exceptions.
>
> The lexical entries are in the format <Concept-Name, Lexicalisation, GrammaticalFeatures> . The grammatical features are a list of attribute-value pairs, e.g., pos – part-of-speech (noun, verb, adjective, etc.), num –

number (singular or plural), massnoun – value is true if this is a mass noun, i.e., uncountable nouns like water.

8.3 Upper Model

Thesauruses have a tree structure relating concepts. This might be useful as way of determining if words refer to the same idea.

Hide me

[6] [7] [3] [1] [9] [16] [2] [17] [11] [12] [13] [4] [14] [15] [18] [19] [20] [21] [5]

Bibliography

1: Kalina Bontcheva, Generating Tailored Textual Summaries from Ontologies, 2005
2: Robert Mac Gregor, Raymond Bates, The Loom Knowledge Representation Language, 1987
3: Kathleen R. McKeown, Discourse Strategies for Generating Natural-Language Text, 1985
4: Mann & Thompson, Rhetorical Structure Theory: A Theory of Text Organisation, 1987
5: Candace Lee Sidner, Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse, 1979
6: Eduard Hovy, Automated Discourse Generation using Discourse Structure Relations, 1993
7: Daniel Marcu, Building Up Rhetorical Structure Trees, 1996
8: Eduard H. Hovy, Natural Language Generation, 1991
9: M.A.K. Halliday, Halliday's Introduction to Functional Grammar r,
10: David Marcu, The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts, 1997
11: Nadjet Bouayad-Agha, Gerard Casamayor and Leo Wanner, Natural Language Generation in the Context of the Semantic Web, 2012
12: William C. Mann, Discourse Structures for Text Generation,
13: Eduard H. Hovy, Approaches to the Planning of Coherent Text, 1988
14: Nick Nicholas, Problems in the Application of Rhetorical Structure Theory to Text Generation, 1994
15: Scott & de Souza, Getting the Message Across in RST-based Text Generation,
16: Bill Mann, An Introduction to Rhetorical Structure Theory, 1999
17: Matthew Stone, etc, Microplanning with Communicative Intentions: The SPUD System, 2001
18: John A. Bateman, Sentence generation and systemic grammar: an introduction, 1997
19: Eduard Hovy and Leo Wanner, Managing Sentence Planning Requirements,
20: Shieber, Stuart M., Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore, Semantic-head-driven generation, 1990
21: Bruce Jakeway and Chrysanne DiMarco, SPLAT: A sentence-plan authoring tool,

Appendix A - Rhetorical Structure Theory

RST is intended to describe texts, rather than the processes of creating or reading and understanding them. It posits a range of possibilities of structure -- various sorts of "building blocks" which can be observed to occur in texts. These "blocks" are at two levels, the principal one dealing with " nuclearity " and " relations " (often called coherence relations in the linguistic literature.) A second level of structures, is called schemas.

While primarily intended to describe entire texts it also applies to paragraphs and sentences.

RST defines four types of defined object: Relations, Schemas, Schema Applications, and Structures.

> Relations are defined to hold between two non-overlapping text spans called the *nucleus* and *satellite*. A relation definition consists of four fields:
>
> 1. Constraints on the nucleus;
> 2. Constraints on the satellite;
> 3. Constraints on the combination of necleus and satellite; and
> 4. The effect; i.e. is it plausible that the specified condition is desired.

The basic idea seems to be that a text can be recursively subdivided into pairs of text spans. The two parts are related by these coherence realtions. Most of the relations are of the form "Nucleus-Satellite" where the Nucleus is the more central to the author's purpose. When the two spans have a similar level of purpose they are termed "Multinuclear".

Relation Name	Nucleus	Satellite
Antithesis	ideas favored by the author	ideas disfavored by the author
Background	text whose understanding is being facilitated	text for facilitating understanding
Circumstance	text expressing the events or ideas occurring in the interpretive context	an interpretive context of situation or time
Concession	situation affirmed by author	situation which is apparently inconsistent but also affirmed by author

Relation Name	Nucleus	Satellite
Condition	action or situation whose occurrence results from the occurrence of the conditioning situation	conditioning situation
Elaboration	basic information	additional information
Enablement	an action	information intended to aid the reader in performing an action
Evaluation	a situation	an evaluative comment about the situation
Evidence	a claim	information intended to increase the reader's belief in the claim
Interpretation	a situation	an interpretation of the situation
Justify	text	information supporting the writer's right to express the text
Motivation	an action	information intended to increase the reader's desire to perform the action
Non-volitional Cause	a situation	another situation which causes that one, but not by anyone's deliberate action
Non-volitional Result	a situation	another situation which is caused by that one, but not by anyone's deliberate action
Otherwise	action or situation whose occurrence results from the lack of occurrence of the conditioning situation	conditioning situation
Preparation	text to be presented	text which prepares the reader to expect and interpret the text to be presented.
Purpose	an intended situation	the intent behind the situation

Relation Name	Nucleus	Satellite
Restatement	a situation	a reexpression of the situation
Solutionhood	a situation or method supporting full or partial satisfaction of the need	a question, request, problem, or other expressed need
Summary	text	a short summary of that text
Volitional Cause	a situation	another situation which causes that one, by someone's deliberate action
Volitional Result	a situation	another situation which is caused by that one, by someone's deliberate action

Multinucleus relations would appear to be of not much use.

Relation Name	Span	Other Span
Contrast	one alternate	the other alternate
Joint	(unconstrained)	(unconstrained)
List	an Item	a next item
Sequence	an item	a next item

The nuclei are essential to the text. Having just the nuclei will leave you with a text that is still coherent. However, the satellites do not form a coherent text.

Each relation has a definition. The definition specifies what a reader of a text must judge to be true in order to include that relation between two spans in an analysis of that text. See Appendix B for the definitions.

Appendix B - RST Definitions

N: Nucleus; S: Satellite; W: Writer; R: Reader.

Relation Name:	Evidence
Constraints on N:	R might not believe N to a degree satisfactory to W.
Constraints on S:	The reader believes S or will find it credible.
Constraints on N+S:	R's comprehending S increases R's belief of N
The Effect:	R's belief of N is increaded.
Locus of Effect:	N
Schema:	(text span N) ← (text span S) [Canonical Order]

Relation Name:	Justify
Constraints on N:	none
Constraints on S:	none
Constraints on N+S:	R's comprehending S increases R's readiness to accept W's right to present N.
The Effect:	R's readiness to accept W's right to present N is increased.
Locus of Effect:	N
Schema:	(text span S) → (text span N) [Canonical Order]

Relation Name: Antithesis

Constraints on N: W has a positive regard for the situation presented in N

Constraints on S: none

Constraints on N+S: The situations presented in N and S are in contrast; because of the incompatibility that arises from the contrast, one cannot have positive regard for both the situations presented in N and S; comprehending S and the incomaptibility between the situations in N and S increases R's positive regard for the situation presented in N.

The Effect: R's positive regard for N is increased.

Locus of Effect: N

Schema: (text span S) → (text span N) [Canonical Order]

Relation Name: Concession

Constraints on N: W has a positive regard for the situation presented in N.

Constraints on S: W is not claiming the situation presented in S doesn't hold

Constraints on N+S: W acknowledges a potential or apparent incompatibility between the situations presented in N and S; W regards the situation presented in N and S as compatible; recognising the compatibility beween the situations presented in N and S increases R's positive regard for the situation presented in N

The Effect: R's positive regard for the situation presented in N is increased.

Locus of Effect: N and S

Schema: (text span S) → (text span N) [Canonical Order]

Relation Name: Circumstance

Constraints on N: none

Constraints on S: S presents a situation (not unrealized)

Constraints on N+S: S sets a framework in the subject matter within which R is intended to interpret the situation presented in N

The Effect: R recognises that the situation presented in S provides the framework for interpreting N

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Solutionhood

Constraints on N: none

Constraints on S: Presents a problem

Constraints on N+S: The situation presented in N is a solution to the problem stated in S

The Effect: R recognises the situation presented in N as a solution to the problem presented in S

Locus of Effect: N and S

Schema: (text span S) → (text span N) [Canonical Order]

Relation Name: Elaboration

Constraints on N: none

Constraints on S: none

Constraints on N+S: S presents additional detail about the situation or some element of subject matter which is presented in N or inferrentially accessible in N in one or more of the ways listed below. In the list if N presents the first member of any pair then S includes the second:

1. set : member
2. abstract : instance
3. whole : part
4. process : step
5. object : attribute
6. generalisation : specific

The Effect: R recognises the situation presented in S as providing additional detail for N. R identifies the element of subjct matter for which detail is provided.

Locus of Effect: N and S

Schema: (text span N) ← (text span S) [Canonical Order]

Relation Name: Background

Constraints on N: R won't comprehend N sufficiently before reading text of S

Constraints on S: none

Constraints on N+S: S increases the ability of R to comprehendan element in N

The Effect: R's ability to comprehend N increases

Locus of Effect: N

Schema: (text span S) → (text span N) [Canonical Order]

Relation Name: Enablement

Constraints on N: Presents R action (including acceptiing an offer), unrealised with respect to the context of N.

Constraints on S: none

Constraints on N+S: R comprehending S increases R's potential ability to perform the action presented in N

The Effect: R's potential ability to perform the action presented in N increases.

Locus of Effect: N

Schema: (text span N) ← (text span S) [Canonical Order]

Relation Name: Motivation

Constraints on N: Presents an action in which R is the actor (including accepting an offer), unrealised with respect to the context of N.

Constraints on S: none

Constraints on N+S: Comprehending S increases R's desire to perform the action presented in N

The Effect: R's desire to perform action presented in N is increased

Locus of Effect: N

Schema: (text span S) → (text span N)

The above two relations can be combined into a single schema:
(text span S) → (text span N) ← (text span S)

Relation Name: Volitional Cause

Constraints on N: Prsents a volitional action or else a situation that could have arisen from a volitional action

Constraints on S: none

Constraints on N+S: S presents a situation that could have caused the agent of the volitional action in N to perform that action;
without the presentation of S, R might not regarf the action as motivated or know the particular motivation;
N is more central to W's purpose in putting forth the N+S combination than S is.

The Effect: R recognises the situation presented in S as a cause for the volition action presented in N

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Nonvolitional Cause

Constraints on N: Presents a situaltion that is not a volitional action

Constraints on S: none

Constraints on N+S: S presents a situation that, by means other than motivating a volitional action, caused the situation presented in N; without the presentation of S, R might not know the particular cause of the situation; a presentation of N is more central than S to W's purposes in putting forth the N+S combination.

The Effect: R recognises the situation presented in S as a cause of the situation presented in N

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Volitional Result

Constraints on N: none

Constraints on S: Presents a volitional action or a situation that could have arisen from a volitional action

Constraints on N+S: N presents a situation that could have caused the situation pesented in S; the situation presented in N is more central to W's purposes that is that presented in S

The Effect: R recognises that the situation presented in N could be a ause for the action presented in S

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Nonvolitional Result

Constraints on N: none

Constraints on S: Presents a situation that is not a volitional action

Constraints on N+S: N presents a situation that caused the situation presented in S; presentation of N is more central to W's purposes in putting forth the N+S combination than is the presentation of S

The Effect: R recognises that the situation presented in N could have caused the situation presented in S

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Purpose

Constraints on N: presents an activity

Constraints on S: presents a situation that is unrealised

Constraints on N+S: S presents a situation to be realised through the activity in N

The Effect: R recognises that the activity in N is initiated in order to realise S

Locus of Effect: N and S

Schema: (text span N) ← (text span S) [Canonical Order]

Relation Name: Condition

Constraints on N: none

Constraints on S: S presents a hypothetical, future, or otherwise unrealised situation (relative to the situational context of S)

Constraints on N+S: Realisation of the situation presented in N depends on realisation of that presented in S

The Effect: R recognises how the realisation of the situation presented in N depends on the realisation of the situation presented in S

Locus of Effect: N and S

Schema: (text span S) → (text span N) [Canonical Order]

Relation Name: Otherwise

Constraints on N: preaents an unrealised situation

Constraints on S: presents an unrealised situation

Constraints on N+S: Realisation of the situationpresented in N prevents the realisation of the situation presented in S

The Effect: R recognises the dependency relation of prevention between the realisationof the situation presented in N and the realisation of the situation presented in S

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Interpretation

Constraints on N: none

Constraints on S: none

Constraints on N+S: S relates the situation presented in N to a framework of ideas not involved in N itself and not concerned with W's positive regard

The Effect: R recognises that S relates the situation presented in N to a framework of ideas not involved in the knowledge presented in N itself

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Evaluation

Constraints on N: none

Constraints on S: none

Constraints on N+S: S relates the situation in N to degree of W's positive regard toward the situation presented in N

The Effect: R recognises that the situation presented in S assess the situation presented in Nand recognises the value it assigns

Locus of Effect: N and S

Schema: (text span S) → (text span N)

Relation Name: Restatement

Constraints on N: none

Constraints on S: none

Constraints on N+S: S restates N, where S and N are of comparable bulk

The Effect: R recognises S as a restatement of N

Locus of Effect: N and S

Schema: (text span N) ← (text span S) [Canonical Order]

Relation Name:	Summary
Constraints on N:	N must be more than one unit
Constraints on S:	none
Constraints on N+S:	S presents a resatement of the content of N that is shorter in bulk
The Effect:	R recognises S as a shorter restatement of N
Locus of Effect:	N and S
Schema:	(text span S) → (text span N)

Relation Name:	Sequence
Constraints on N:	multi-nuclear
Constraints on S:	
Constraints on N+S:	A succession relationship between the situations is presented in the nuclei
The Effect:	R recognises the succession relationship among the nuclei
Locus of Effect:	multiple nuclei
Schema:	(text span) - (text span) - (text span)

Relation Name:	Contrast
Constraints on N:	multi-nuclear
Constraints on S:	
Constraints on N+S:	no more than two nuclei; the situations presented in these two nuclei are (a) comprehended as the same in many respects, (b) comprehended as different in a few respects, and (c) compared with respect to one or more of these differences.
The Effect:	R recognises the comparability and the differences yielded by the comparison is being made
Locus of Effect:	multiple nuclei
Schema:	(text span N) ↔ (text span N)

Appendix C - Cue Phrases

Cues that show addition and introduce examples:

Again	Also	Lastly	To illustrate
And	Incidentally	Finally	In other words
And then	Moreover	What’s more	Hence
Besides	Nor	In addition	First, Second, Third, etc.
Equally important	Too	For example	Next
Further	Furthermore	To demonstrate	For instance

Cues that emphasize:

After all	That is	As I have said	Unquestionably
Obviously	Above all	As I have noted	In brief
In fact	Of course	As noted	In short
As amatter of fact	Again	In any case	To be sure
Indeed	To repeat	In any event	

Cues that introduce conclusions or summarize:

Hence	In brief	Therefore	On the whole
Consequently	Summing up	Thus	To conclude
As a result	In conclusion		

Cues that make the reader stop and compareor contrast:

But	At the same time	And yet	Still
Notwithstanding	Although this is true	On the contrary	However
On the other hand	Conversely	After all	Yet
Meanwhile	For all that	Likewise	Nonetheless
Nevertheless	In the same manner	In comparison	Although
In contrast	Simultaneously	While this is true	By and large
In the same way	Equally	In spite of	Similarly

Cues thatshow cause and effect:

And so	Consequently	On account of	Accordingly
Due to	Since	As a result	Hence
Therefore	Because of	If	Thus

Cues that show time:

Immediately following	Later	At times	Presently
In the future	At length	Afterwards	Currently
Earlier	Subsequently	Before	Soon
First, Second, etc.	Meanwhile	Finally	Soon after
Previously	Next	From now on	Eventually
Immediately thereafter	Once	During	Then

Appendix D - An Experiment with RST

The experiment was to try to write down what the data set would look like after each step of a process that transformed RDF into English using Rhetoriacal Structure Theory.

```
//      An experiment in Rhetorical Structure Theory

// The following are the concept texts for each initial sentence.

1. system allows user
2. user uses system
3. request is created by user
4. request has viewpoints
5. document is based on project ontologies
6. document is based on SASSY ontologies
7. document presents views
8. action is supported by system
9. action is intiated by request
10. action generates architecture document

//=============================================================
// Add correction for views.
//      This should be a correction to the actual RDF

1. system allows user
2. user uses system
3. request is created by user
4. request has viewpoints
5. views are based on project ontologies
6. views are based on SASSY ontologies
7. document presents views
8. action is supported by system
9. action is intiated by request
10. action generates architecture document

//=============================================================
// Reverse passive form statements

1. system allows user
2. user uses system
3. request is created by user
3. user creates request
4. request has viewpoints
5. views are based on project ontologies
6. views are based on SASSY ontologies
7. document presents views
8. action is supported by system
8. system supports action
9. action is intiated by request
9. request initiates action
10. action generates architecture document

//=============================================================
// Aggregate 5 and 6 as they have same subject and predicate

1. system allows user
2. user uses system
3. user creates request
4. request has viewpoints
5. views are based on project and SASSY ontologies
6. document presents views
7. system supports action
8. request initiates action
9. action generates architecture document
```

```
//============================================================
// Group based on common subject or object.
//
// This hypothesis is based on the observation that if there is no common
// subject or object then it is quite difficult to suggest any relation that
// could link the two clauses.

System
1. system allows user
2. user uses system
7. system supports action

User
1. system allows user
2. user uses system
3. user creates request

Request
3. user creates request
4. request has viewpoints
8. request initiates action

Views
5. views are based on project and SASSY ontologies
6. document presents views

Document
6. document presents views
9. action generates architecture document

Action
7. system supports action
8. request initiates action
9. action generates architecture document

//============================================================
// Create pairs and remove duplicates

System
        system allows user
        user uses system

        system allows user
        system supports action

        user uses system
        system supports action

User
        system allows user
        user creates request

        user uses system
        user creates request

Request
        user creates request
        request has viewpoints

        user creates request
        request initiates action

        request has viewpoints
        request initiates action

Views
        views are based on project and SASSY ontologies
        document presents views

Document
        document presents views
        action generates architecture document
```

```
Action
        system supports action
        request initiates action

        system supports action
        action generates architecture document

        request initiates action
        action generates architecture document

//============================================================
// Assign RST relations
//      Not very clear on how to automate this.
//      Perhaps there might be a mapping from a thesaurus classification
//      to the RST relation. The originators of RST studiously avoided
//      anything that could be automated.

System
        Condition
        N: 2. user uses system
        S: 1. system allows user

        Condition
        N: 7. system supports action
        S: 1. system allows user

        Condition
        N: 7. system supports action
        S: 2. user uses system

User
        Condition
        N: 3. user creates request
        S: 1. system allows user

        Condition
        N: 3. user creates request
        S: 2. user uses system

Request
        Elaboration
        N: 3. user creates request
        S: 4. request has viewpoints

        Sequence
        N 1: 3. user creates request
        N 2: 8. request initiates action

        Elaboration
        N: 8. request initiates action
        S: 4. request has viewpoints

Views
        Elaboration
        N: 6. document presents views
        S: 5. views are based on project and SASSY ontologies

Document
        Elaboration
        N: 9. action generates architecture document
        S: 6. document presents views

Action
        Condition
        N: 8. request initiates action
        S: 7. system supports action

        Purpose
        N: 9. action generates architecture document
        S: 7. system supports action

        Sequence
        N 1: 8. request initiates action
```

```
        N 2: 9. action generates architecture document


//============================================================
// Construct RST Tree

Purpose
S: 7. system supports action
N: 9. action generates architecture document

Sequence
N 1: 3. user creates request
        Condition
        N: 3. user creates request
                Elaboration
                N: 3. user creates request
                S: 4. request has viewpoints
        S: 2. user uses system
                Condition
                N: 2. user uses system
                S: 1. system allows user
N 2: 8. request initiates action
N 3: 9. action generates architecture document
        Elaboration
        N: 9. action generates architecture document
        S: 6. document presents views
                Elaboration
                N: 6. document presents views
                S: 5. views are based on project and SASSY
        ontologies



//============================================================
// Aggregation etc

system supports generates architecture document

system allows user to use the system to create request from viewpoints

request initiates generates architecture document that presents views     based
on project and SASSY ontologies.


//============================================================
// From NLG

System supports the generation of architecture documents.

The system allows a user to use the system to create a request from viewpoints.

The request initiates the generation of architecture documents that present
views that are based on project and SASSY ontologies.

//============================================================

The SASSY system shall support the generation of architecture documents. The
system shall allow a user to use the system to create a request from
viewpoints. Then, the request will initiate the generation of architecture
documents that present views that are based on project and SASSY ontologies.

//============================================================
```

Appendix E - An Experiment Using Graph Theory

The experiment was to try to write down what the data set would look like after each step of a process that transformed RDF into English using Graph Theory to identify a backbone for the graph.

```
// ============================================================
// Start with some RDF  - duplicates are for reversed predicates
// where the passive voice has been replaced. This could be handled
// by a reasoner.

1. system allows user
2. user uses system
3. request is created by user
3. user creates request
4. request has viewpoints
5. views are based on project ontologies
6. views are based on SASSY ontologies
7. document presents views
8. action is supported by system
8. system supports action
9. action is intiated by request
9. request initiates action
10. action generates architecture document

// ============================================================
// Put into linked order

1. system allows user
2. user uses system
3. user creates request
4. request has viewpoints
5. request initiates action
6. system supports action
7. action generates achitecture document
8. document presents view
9. views are based on project ontologies
10. views are based on SASSY ontologies

// ============================================================
// The backbone is the following set:

1. system allows user
3. user creates request
5. request initiates action
7. action generates achitecture document
8. document presents view
9. views are based on project ontologies

// ============================================================
// Statements 6 and 7 are directly linked [good coherence] and are
// important according to the context.

6. system supports action
7. action generates achitecture document

// They can be combined into a sentence that summarises the paragraph.
6. system supports generation of architecture document

// ============================================================
// Statement 9 and 10 share subject and predicate and can be aggregated

9. views are based on project and SASSY ontologies

// ============================================================
// The predicates "creates", "initiates", "generates" are
// linked and have a temporal aspect to their meaning. This can
// be used to create a sequential story line for the paragraph.

1. system allows user
```

```
2. user uses system
3. user creates request
5. request initiates action
7. action generates achitecture document

// ============================================================
// The predicates "has", "presents", "is based on" are, to use
// an RST term, indicative of elaborations.

3. user creates request
4. request has viewpoints

7. action generates achitecture document
8. document presents view
9. views are based on project and SASSY ontologies

// ============================================================
// Combining 5 with 7, 8 & 9 is at the limits of complexity.

5. request initiates action
7. action generates achitecture document
8. document presents view
9. views are based on project and SASSY ontologies

// ============================================================
// We now have the following:

6. system supports action
7. action generates achitecture document

1. system allows user
2. user uses system
3. user creates request
4. request has viewpoints

5. request initiates action
7. action generates achitecture document
8. document presents view
9. views are based on project and SASSY ontologies

// ============================================================
// Form in to sentences. "Action" is an implicit object that
// was introduced into the RDF to handle a complex predicate.

6. system supports generation of achitecture document

1. system allows user to use system to create request that has viewpoints.

5. request initiates generation of achitecture document
8. that presents view based on project and SASSY ontologies

// ============================================================
// Add in articles and proper names

The SASSY system supports the generation of achitecture documents.

The system allows a user to use system to create requests that have viewpoints.

The request initiates the generation of achitecture documents
that present views based on the project and SASSY ontologies.


// ============================================================
// Add a cue phrase to the last sentence to help convey the
// sequential aspect. Also, based on the context data, set the
// tense to future and use "shall" as a modal.

The SASSY system shall support the generation of achitecture documents.

The system shall allow a user to use system to create requests
which have viewpoints.

Then the requests will initiate the generation of achitecture documents
```

```
that present views based on the project and SASSY ontologies.


// ============================================================
// Finally the resultant paragraph:

The SASSY system shall support the generation of achitecture
documents. The system shall allow a user to use system to create requests that
have viewpoints. Then, the requests will initiate the generation of achitecture
documents that present views based on the project and SASSY ontologies.

// ============================================================
```