Step by step: underspecified markup in incremental rhetorical analysis [*]

David Reitter
Media Lab Europe
Adaptive Speech Interfaces
Sugar House Lane, Dublin 8, Ireland
reitter@mle.media.mit.edu

Manfred Stede
University of Potsdam
Dept. of Linguistics
P.O. Box 601553, 14415 Potsdam, Germany
stede@ling.uni-potsdam.de

Abstract

While quite a few linguistic corpora with syntactic annotations are available today, resources are scarce on the level of discourse annotation. As a step toward flexible, extendible *annotation schemes*, we propose an XML format for annotating rhetorical structure trees. For human and automatic analysis alike, rhetorical structure is often difficult and assigned incrementally. Thus, our format allows for *underspecification*. The paper discusses the various design decisions involved, illustrates the format with an example, and sketches some applications.

1 Introduction

The demand for annotated linguistic corpora rises steadily. Recently, work has begun on providing data that is annotated not only on the sentence level but also on the discourse level. In particular, 'rhetorical annotation' turned out to be important for applications such as automatic summarization. With the growing importance of machine learning and parsing algorithms that detect rhetorical structure and evaluate the results of manual or automatic annotation, it becomes clear that corpus data should be readable for both humans and machines.

In the following, we discuss design considerations for an XML-based rhetorical annotation format that is extensible and provides room for underspecification. This is a key feature for two reasons: 1) Human annotators often find it difficult to make a clear decision on either a specific relation or the length of the spans (e.g., in the case of sentences starting with a conjunctive adverbial: *On the other hand, ...*). Thus it should be possible to leave such a matter open and represent it accordingly. 2) For automatic rhetorical analysis, the problem above is much more pressing. Rather than enforcing a decision all the time, it is desirable for a parser to leave some aspects underspecified, represent this clearly, and possibly have additional components making a choice later on the basis of additional knowledge. More generally, underspecification allows for *incremental* rhetorical analysis based on sound representations. We assume a theory of rhetorical description along the lines of Mann & Thompson (1988), which assigns relations between adjacent spans of text and recursively builds up a tree. We have applied our format to both manual and experimental automatic analyses of a new corpus of German newspaper texts. The format is open to extension and specialization, e.g. to enable multi-modal applications. In general, we think that the emergence of a standard for rhetorical annotation will be instrumental for comparing analyses, and obviously for training stochastic or machine learning algorithms.

2 Previous work

Schilder (2002) formalizes a symbolic underspec-

[*]The work presented was funded in part through EU grant IST-2001-38685. We would like to thank Silvan Heintze for his helpful comments.

ification scheme for rhetorical structure. His relations connect Segmented Discourse Representation Structures; he explicitly states immediate dominance, dominance, precedence and equivalence of text spans. Similar to our approach, a set of relations is given for the whole document, and for a specific pair of segments, the set of relations may be constrained during one of the analysis steps. His symbolic system is targeted at an analysis architecture based on cue phrases and a topicality measure that both constrain the rhetorical structure. In contrast, our architecture defines a serialized, XML based intermediate format that allows for the exchange of corpus data and for an *incremental* annotation of cues and rhetorical structure. Other representation efforts include Rehm (1998), who uses an SGML syntax to identify rhetorical cues. The annotation application RSTTool (O'Donnell 2000) writes annotation data from single documents in several SGML based formats that include no underspecification. In contrast to this format, our representation integrates various layers of annotation; this is targeted primarily at automatic analysis, which will have to take into account quite different kinds of information before making a decision about rhetorical relations (Marcu 2000, Corston-Oliver 1998).

3 Accounts of rhetorical structure

3.1 Basis: Rhetorical Structure Theory

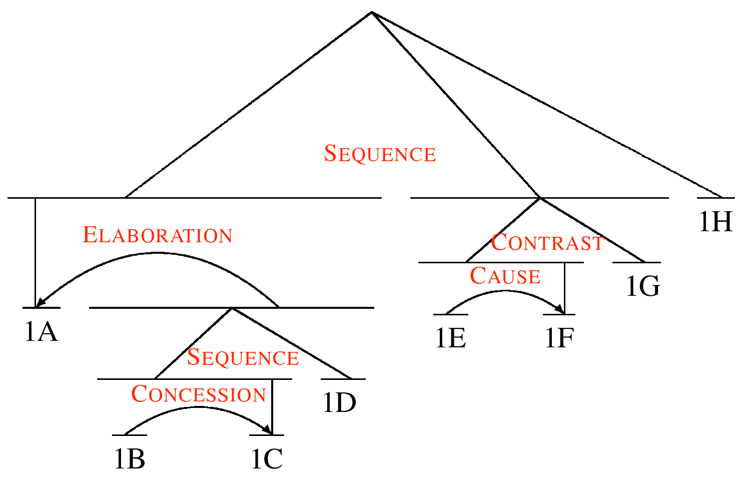
Rhetorical Structure Theory (Mann & Thompson 1988, RST) builds tree structures on top of rhetorical relationships that hold among minimal units of text and recursively among larger units as well. RST proposes a set of such relations, including e.g. ELABORATION(A, B), which is defined as a text span B giving additional information regarding the facts presented in span A, and CONTRAST, which is defined as the contents of two or more text spans being knowingly presented as incompatible. Depending on the relation, a text span may take the role of *nucleus* or *satellite*. 'Paratactic' relations hold among several nuclei; 'hypotactic' relations hold between one nucleus and one satellite. In fig. 1, satellites are designated by arcs, nuclei by straight lines.

The relations in RST were chosen such that the resulting tree is a characterization of a text's overall coherence. Their definitions are not linked to linguistic properties such as connectives or sentence types. Semantic and world-knowledge based constraints are supposed to interact with the presumed writer's or speaker's intentions. The set of relations was proposed on the basis of empirical work with a wide variety of texts; RST does not claim that this set be "closed". As for the span structure, RST posits that relations do not overlap and do not share elements (except hypotactic relations sharing a nucleus — see below) . Marcu (2000) provides a formalization of the framework of rhetorical structure.

3.2 Binary trees

The proposal by Mann & Thompson (1988) offers the possibility of binary hypotactic relations sharing a common nucleus. Each nucleus and all its satellites form the instantiation of a *schema*. Subsequent applications of the theory, however, mainly made use of the idea of binary trees (Cristea & Webber 1997, Marcu 2000, and others). These contain a nucleus and exactly one satellite, or exactly two nuclei, respectively.

In Fig. 2, analysis (1) can be represented as shown in analysis (2). This assumes that the nucleus contributes the essential meaning in comparison to its satellite and "promotes it upward". Consequently, if a relation holds between span A with several segments and span B, the same (or, a similar) relation holds between the nucleus of span A and span B (Marcu 2000, compositionality criterion). Empirical evidence we gained in the collection of our newspaper corpus supports this view: in the intuition of the annotators, in no case was the non-nucleus-sharing, tree-structured representation representationally weaker than the nucleus-sharing view. The choice between the two possibilities for a two-satellite-one-nucleus schema was usually made from referential clues. Also, in the RST Discourse Treebank (Carlson et al. 2001), we found that only in 9 out of 17962 relation schemas (in the partition labeled 'training') there were satellites relating to a common nucleus with a hypotactic relation. We therefore chose to implement just binary relations and no RST schemas.



[Yesterday, the delegates chose their new representative.]^{1A} [Even though Smith received only 24 votes,]^{1B} [he accepted the election with a short speech.]^{1C} [Then the assembly applauded for three minutes.]^{1D} [Due to the upcoming caucus meeting,]^{1E} [the subsequent discussion was very short.]^{1F} [Nonetheless the most pressing questions could be resolved.]^{1G} [The meeting was closed at 7pm.]^{1H}

Figure 1: A text analysis within Rhetorical Structure Theory. It is one of the interpretations that can be derived from the underspecified URML representation in Fig. 3

3.3 Is a single tree adequate?

The tree-structured approach of RST prima facie suggests that there is – even though several distinct analyses may be drawn from a text (Mann & Thompson (1988) themselves had acknowledged potential sources of ambiguity) – only one structure that is faithful to the writer's intentions. It may however be questioned whether there is indeed such a "primary rhetorical intention" (Grosz & Sidner 1986). Furthermore, when it comes to automatic rhetorical analysis, a program will have difficulties deriving the one and only correct structure. One of our own approaches (Reitter to appear) tries to derive a single interpretation that *most likely* matches the writer's intention; however, the algorithm can also output several interpretations, along with their probabilistic scores as derived from corpus training. — In conclusion, we propose that a representation format should provide the possibility of multiple analyses.

4 Rhetorical annotation in URML

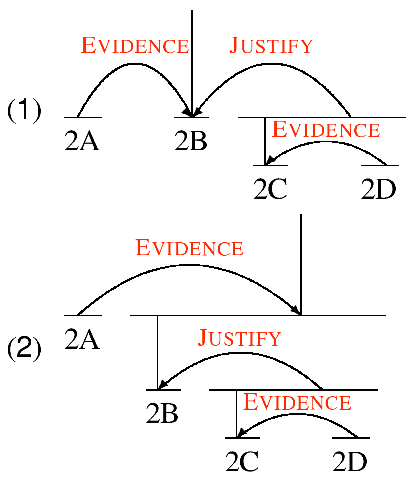
4.1 Basics

Our representation format URML (Underspecified rhetorical markup language) allows for a free definition of dependencies among text spans. It clearly separates: 1) Symbolic system and, in the document, an inventory of relation instances. A document annotation contains a set of relations that can be found in the data. Similar to a parse forest in syntactic parsing (Billot & Lang 1989), it may state ambiguous relations. Some may even be incompatible according to the axioms that defined a well-formed rhetorical analysis. This is defined by the format; all relation instances may be represented using URML. 2) Axioms and, in the document, well-formed tree analyses of a text. It is up to the client software to define restrictions that hold for well-formed analyses. A well-formed analysis contains a subset of the relations defined in the relation set of the document. This well-formed analysis can also be represented and identified as such in URML. We chose XML as underlying formalism in order to maximise re-usability. A "document-type definition" grammar was defined to describe the format of documents, rhetorical and morpho-syntactic annotations.[1]

In URML, all documents are contained in one file. This facilitates automatic handling, since the filesystem is not involved in retrieving parts of the corpus. To make use of the XML-based data, standard XML libraries are available for all common implementation formalisms.

Relation names are declared once for all documents (Fig. 3, `<reltypes>`). Individual documents consist of minimal discourse

[1]The complete document type defintion (DTD) for URML is available from our web site, `http://www.ling.uni-potsdam.de/cl/rst/`



[Also, there can be no social boundaries to the seldom-publicized topic of a community-level obligation to social welfare.]^{2A} [The moral responsibility is ignored.]^{2B} [The question is, whether the disadvantaged group of the homeless is put away even more in their new quarters in Kyritz.]^{2C} [After all, their home is Wittstock.]^{2D} (maz14071e)

Figure 2: Schemas vs. binary nodes. Text translated from original.

units (`<text>`), followed by relation nodes (`<analysis>`) explained below.

When analysis information is added to the raw data, we want to preserve the original information wherever possible (especially in the light of the incremental rhetorical parsing we are developing; see below). Thus, all information used in the analysis process is stored in the corpus persistently.

4.2 Meta-data

The possibility to trace partial analyses of documents is valuable, especially in large corpora. For example, when a manually annotated rhetorical relation is to be clarified later, there should be a way to identify the annotator. Similarly, documents may get corrupted due to bugs in an annotation tool. Thus, a source identifier is needed. Within an `<info>` tag, source information for documents and a reference to the annotator can be given.

```
<analysis id="maz3379.a.1" scheme=
                             "interpretation">
 <info>
```

```
 <editor job="annotate" date="18.02.02">
    Antje Sauermann
 </editor>
 <editor job="revise" date="20.09.02">
    David Reitter
      <note>revised result/cause
            nuclearity
      </note>
...
```

4.3 Representing tree structure in XML

One crucial decision in syntax design was whether annotations should be coded as on-the-spot markup within the text or as relation nodes with referential indices. The first variant is easy to read (and is analogous to the one used in a LISP-style format in Carlson et al. (2001)). For example:

```
<concession>
  <satellite>Admittedly, ...</satellite>
       <nucleus> <contrast>
  <nucleus>However, some ...</nucleus> <
      nucleus>others ...</nucleus>
  </contrast> </nucleus> </concession>
```

This format demands a fully specified, unambiguous single tree structure and encodes the underlying relation set in the document grammar (DTD). In XML, this may be desirable when generic editors are used, because they restrict annotators to comply with the DTD, thus with claims of the underlying discourse theory. All processing modules, however, are also determined to follow the fixed syntax. Changes in theoretic assumptions, such as the introduction of discontinuous constituents or several rhetorical analysis layers, inevitably lead to a chain of modifications in the system, even if the changes concern only one analysis layer.

Therefore, in our format every node of a discourse tree represents one relation: either a hypotactic one (one nucleus, one satellite) or a paratactic one (several nuclei). Nodes are indexed and reference each other to express references to the according text spans. In the following example, the IDs 1, 2, 3 refer to minimal discourse units (cf. Fig. 3).

```
<hypRelation type="concession" id="10">
        <satellite id="1" />
        <nucleus id="11" />
</hypRelation>
<parRelation type="contrast" id="11">
```

```
        <nucleus id="2" />
        <nucleus id="3" />
</parRelation>
```

Each relation statement refers to its direct descendants in the tree via the identifier of the relation. We could, alternatively, refer to spans by their left and right borders in the sequence of minimal discourse units. This way, ambiguous analyses would share common nodes high up in the tree. However, in a scenario of probabilistic analysis, it would violate our assumption that the score assigned to those nodes relates to their descendants. In the example given in Fig. 1, the score given to the CAUSE relation might be lower because the connective *then* in segment 1D usually indicates a SEQUENCE relation.

Paratactic and hypotactic relations are represented in the same way, and there are no additional span types. In contrast to the RSTTool format (O'Donnell 2000), our URML format is not aimed at presenting and manipulating RST *diagrams*. Rather, we wish to store rhetorically annotated data, potentially underspecified, in a manner that is independent from a particular application and readable for both humans and machines.

In contrast to semantics-based representations (Schilder 2002), URML refers to textual data. It implicitly states linear precedence. Tree nodes in an analysis state immediate dominance; underspecified dominance situations have to be explicitly stated with concurrent tree nodes. This keeps annotations simple enough to work with them both manually and automatically.

4.4 Underspecification

Consider again the sample text shown in Fig. 1. What is its primary discourse intention, and what structure should be ascribed? Annotators may disagree. For example, 1D may be rightly characterized as being in temporal sequence with span [1B,1C], but it could also seen as in sequence with only 1C, with 1B being a concession to span [1C,1D]. Automated analysis tools might only give a partial answer here. Also, they might not be able to infer the ELABORATION relation between 1A and the subsequent segments. In particular, automatic analysis will often encounter problems to locate the precise boundaries of larger segments: Where does the just-mentioned ELABORATION end? Also, *nonetheless* at the beginning of 1G signals a CONTRAST or CONCESSION, but based solely on surface cues, it is by no means clear how far to the left the first span stretches, i.e., what the exact scope of the *nonetheless* is.

The referential markup syntax is flexible by design. Uncertainties on the kind of relation can be represented by simply leaving out relation information – see Fig. 3. For instance, if the specific relation between two spans is unknown, the `relation` tag can omit the `type` attribute (node12). If, however, the class of a relation (hypotactic or paratactic) is known, a `hypRelation` or `parRelation` tag should be used. If a span is known to be an argument of a relation, but its role is unknown, it should be labeled `element` instead of `satellite` or `nucleus` (nodes 1E, 1F). Also, *scores* may be mentioned for a node to indicate a preference or the result of some heuristics. As for structural underspecification, the set of relations implements what is known in chart parsing systems as *subtree sharing*: We do not represent each tree derivation separately, but several analyses may share sub-trees. This happens when two or more alternative relations refer to the same relation as their nucleus or satellite (node10a and node10b). Another technique allows structure-sharing of nodes that are the same, but have different subtrees (*local ambiguity packing*). To do this, we introduce an attribute `group` which defines a common name (node10) for two or more alternative relations. Node11 refers to either one of the three subtrees (with node10a/b/c). While local ambiguity packing saves space, it is limited to those cases where the unified node contains exactly the same data. This applies to the `score` given to node14 which might depend on the (ambiguous) structure of its content. If so, local ambiguity packing cannot be used by the client application.

In case of dependencies among alternative nodes, separate analyses should be used. The rationale for this suggestion is a lower complexity at the client application side. The alternative, an explicit disjunction with a tag that groups relations within the relation set, would demand more elaborate processing of the representation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE urml SYSTEM "urml.dtd">
<urml>
 <header>
  <reltypes>
   <rel name="Cause" type="hyp"/>
   <rel name="Circumstance" type="hyp"/>
   <rel name="Concession" type="hyp"/>
   <rel name="Condition" type="hyp"/>
   <rel name="Contrast" type="par"/>
   <rel name="Elaboration" type="hyp"/>
   <rel name="Joint" type="par"/>
   <rel name="List" type="par"/>
   <rel name="Means" type="hyp"/>
   <rel name="Purpose" type="hyp"/>
   <rel name="Result" type="par"/>
   <rel name="Sequence" type="par"/>
  </reltypes>
 </header>
 <document id="sample001">
  <text>
   <segment id="1A">Yesterday, the
       delegates
       chose their new
       representative.</segment>
   <segment id="1B">Even though Smith
       received only 24 votes,</segment>
   <segment id="1C">he accepted the
       election with a short
       speech.</segment>
   <segment id="1D">Then the assembly
       applauded for three
       minutes.</segment>
   <segment id="1E">Due to the upcoming
       caucus meeting,</segment>
   <segment id="1F">the subsequent
       discussion was very
       short.</segment>
   <segment id="1G">Nonetheless the most
       pressing questions could be
       resolved.</segment>
   <segment id="1H">The meeting was closed
       at 7pm. <P /></segment>
 </text>
  <analysis status="forest-complete">
  <hypRelation id="node9a" type="Concession">
     <satellite id="1B"/>
     <nucleus id="1C"/>
  </hypRelation>
  <parRelation id="node9b" type="Sequence">
     <nucleus id="1C"/>
         <nucleus id="1D"/>
  </hypRelation>
  <hypRelation id="node10a" group="node10"
      type="Cause" score=".3">
     <satellite id="node9a"/>
     <nucleus id="1D"/>
  </hypRelation>
  <parRelation id="node10b" group="node10"
      type="Sequence" score=".6">
     <nucleus id="node9a"/>
     <nucleus id="1D"/>
  </hypRelation>
  <hypRelation id="node10c" group="node10"
      type="Concession" score=".1">
     <satellite id="1B"/>
     <nucleus id="node9b"/>
  </hypRelation>
  <hypRelation id="node11" type="Elaboration"
       score=".4">
     <nucleus id="1A"/>
     <satellite id="node10"/>
  </hypRelation>
  <relation id="node12">
     <element id="1E"/>
     <element id="1F"/>
  </hypRelation>
  <parRelation id="node13" type="Contrast">
     <nucleus id="node12"/>
     <nucleus id="1G"/>r
  </parRelation>
  <parRelation id="node14" type="Sequence">
     <nucleus id="node11"/>
     <nucleus id="node13"/>
     <nucleus id="1H"/>
  </parRelation>
  </analysis>
 </document></urml>
```

Figure 3: Sample URML document. Meta data is not given.

4.5 Interpreting relation sets

Turning now to automatic tools for structural analysis, we encounter three paradigms to understand sets of relations, which correspond to different phases of analysis. The first is the *parse forest* scheme, where concurrent partial analyses are present. The parse forest holds relations already processed. They may be annotated with a score (Fig. 3, node10a/b/c). We indicate the scheme in the `<analysis>` element with a `status="forest"` attribute.

Different stages of discourse analysis will modify existing relation scores and add new relations. At this phase of analysis (`status ="forest-complete"`), a missing relation in the URML document indicates that this relation has not been considered yet. If it was considered, it should be included, possibly with a `score="0"` attribute. When the process is finished, pruning may occur and low-scoring relations may be removed from the relation set. This changes the semantics of the parse forest: relations that don't exist in the forest are assumed not to hold.

The third way to see a set of relations is the `interpretation` scheme. Here, the analysis algorithm has singled out (and, possibly, scored) a whole, well-formed derivation.

As shown, the *forest* scheme assigns scores to each relation, while the second scheme assigns them to each analysis. This happens on grounds of the locality of classification decisions. In bottom-up style algorithms, the partial analyses are preselected only according to local constraints, i.e. constraints that refer to data covered by the local text spans. For example, the rhetorical relation LIST might be proposed to hold between the segments B and C, because of a comma found at the right border of B and the connective *and* found at the left border of C. At a later stage in processing, the analysis algorithm might find that B, C elaborate on A, which ends with the words "is highly contradictory" and a colon. It may revise its earlier decision and find that a CONTRAST relation holds between B, C, because of the ELABORATION relation and the cue phrase to be found. These properties are *non-local.*

The three-scheme layout shown here does not restrict analysis algorithms to work decrementally, reducing the search-space tool by tool. They can, alternatively, add anticipated relations. These tools should add their own `<analysis>` to the document.

4.6 Specializing the DTD

The proposed document type definition (DTD) is open. We see it as a base class that may be extended. Therefore, unknown tags should be ignored by applications. We have used a derived DTD which provides optional part-of-speech information for each token of a text and includes results from a stemming algorithm. Other extensions could, e.g., designate boundaries of topic chains or disambiguate discourse markers. Existing documents should still be valid with a specialized DTD, so that corpus data can be imported for evaluation. For exchanging data in the other direction, additional tags can usually be easily stripped away.

5 Some applications

5.1 Collecting rhetorical corpora with URML

The format described has evolved from a practical application. We collected a corpus of German language newspaper texts and performed manual rhetorical annotation confirming to RST. A web-crawler proved to be the most practical solution to download and normalize the newspaper texts from a web site. With the consent of the publisher, it accessed the newspaper's online edition, stripped away layout-specific markup and conserved exact source information including some meta-data regarding authorship, date, newspaper section etc.

Two annotators worked through 173 texts. Data was converted from the annotation application format to URML, part-of-speech-tagged and segmentized with Perl tools. These access external tokenization and tagging applications. We used a GUI-based tool (O'Donnell 2000, RSTTool 3.1) to annotate the data with rhetorical structures and then converted it to URML.

Also, the RST-annotated collection of English language newspaper articles presented in Carlson et al. (2001) was converted to URML. It contains almost 21,800 minimal discourse units in 385 documents of varying sizes.

For both corpora, URML can represent all of the structures.

5.2 Learning classification with URML

We also used the underspecified representation in a machine-learning approach to rhetorical analysis (Reitter to appear). Our rhetorical classification decisions are based on a variety of shallow features, which are annotated by a chain of tools. Our machine learning tool uses a standard XML parser (Xerces) to build a DOM representation and operates on this data in memory. As the further feature extraction process is costly in terms of space and time, we use indexing algorithms to access data quickly. At any time, we can serialize the current parse forest or single relations, which provides an adequate debugging and optimization output.

5.3 Building tools with XML/DOM

We use the data format to incrementally annotate data using various tools. The XML-based format serves as common interface between the tools. This interface standardization facilitates the implementation of a layer-based processing model. Each layer may be assigned to different machine or human annotators.

The Document Object Model (DOM) defines an interface to access a tree-structure of the document, created by an off-the-shelf XML parser. This allows fairly easy access to the data. The development of annotation tools profited from the fact that corpus-related DOM data could be visualized easily through a generic formating routine at any point during a system run.

For simple tools that access only partial content, such as tokenizers and POS taggers, we recommend to have them operate on the raw XML files with regular expressions.

5.4 Editing and Visualization

Thanks to the XML architecture, the URML-based corpus can be inspected and manually edited with one of numerous XML browsers and editors available. For the purpose of visualization, we provide a package for LaTeX(Reitter 2002). Rhetorical analyses and their corresponding document text can be extracted from the URML data and converted to the appropriate format.

6 Conclusion

We have shown an underspecification method for rhetorical structure and introduced an XML-based corpus format. The format has already proven to be useful in corpus collection efforts, in a pipeline-based rhetorical parser, and in the implementation of a machine-learning analysis algorithm. We would like to see it as proposal towards a standardization of corpus representation and for tool building in rhetorical analysis.

References

Billot, S. & Lang, B. (1989), The structure of shared forests in ambiguous parsing, *in* 'Proceedings of the 27th Meeting of the Association for Computational Linguistics', pp. 143–151.

Carlson, L., Marcu, D. & Okurowski, M. E. (2001), Building a discourse-tagged corpus in the framework of rhetorical structure theory, *in* 'Proc. of the 2nd SIGDIAL Workshop on Discourse and Dialogue, Eurospeech', Denmark.

Corston-Oliver, S. H. (1998), Computing Representations of Discourse Structure, PhD thesis, University of California, Santa Barbara, CA.

Cristea, D. & Webber, B. (1997), Expectations in incremental discourse processing, *in* 'Proc. of ACL-EACL97', Madrid, Spain, pp. 88–95.

Grosz, B. J. & Sidner, C. L. (1986), 'Attention, intentions, and the structure of discourse', *Computational Linguistics* **12**, 175–204.

Mann, W. C. & Thompson, S. A. (1988), 'Rhetorical Structure Theory: Towards a functional theory of text organization', *Text* **8**(3), 243–281.

Marcu, D. (2000), *The theory and practice of discourse parsing and summarization*, MIT Press, Cambridge, MA.

O'Donnell, M. (2000), RSTTool 2.4 – a markup tool for Rhetorical Structure Theory, *in* 'Proc. of the 1st International Natural Language Generation Conference, Mitzpe Ramon, Israel'.

Rehm, G. (1998), Vorüberlegungen zur automatischen Zusammenfassung deutschsprachiger Texte mittels einer SGML- und DSSSL-basierten Repräsentation von RST-Relationen, Master's thesis, Universität Giessen.

Reitter, D. (2002), Rhetorical theory in LaTeX with the 'rst' package, Technical report, http://www.reitter-it-media.de/compling/.

Reitter, D. (to appear), Complex signals for rhetorics: On rhetorical analysis with rich-feature support vector models, *in* 'in Proceedings of the GLDV conference 2003'.

Schilder, F. (2002), 'Robust discourse parsing via discourse markers, topicality and position', *Natural Language Engineering* **8** (2/3).