A Dependency Perspective on RST Discourse Parsing and Evaluation

Mathieu Morey
IRIT, Université Toulouse
CNRS, Université Paul Sabatier
mathieu.morey@irit.fr

Philippe Muller
IRIT, Université Toulouse
CNRS, Université Paul Sabatier
philippe.muller@irit.fr

Nicholas Asher
IRIT, Université Toulouse
CNRS, Université Paul Sabatier
nicholas.asher@irit.fr

Computational text-level discourse analysis mostly happens within Rhetorical Structure Theory (RST), whose structures have classically been presented as constituency trees, and relies on data from the RST Discourse Treebank (RST-DT); as a result, the RST discourse parsing community has largely borrowed from the syntactic constituency parsing community. The standard evaluation procedure for RST discourse parsers is thus a simplified variant of PARSEVAL, and most RST discourse parsers use techniques that originated in syntactic constituency parsing. In this article, we isolate a number of conceptual and computational problems with the constituency hypothesis.

We then examine the consequences, for the implementation and evaluation of RST discourse parsers, of adopting a dependency perspective on RST structures, a view advocated so far only by a few approaches to discourse parsing. While doing that, we show the importance of the notion of headedness of RST structures. We analyze RST discourse parsing as dependency parsing by adapting to RST a recent proposal in syntactic parsing that relies on head-ordered dependency trees*, a representation isomorphic to headed constituency trees. We show how to convert the original trees from the RST corpus, RST-DT, and their binarized versions used by all existing RST parsers to head-ordered dependency trees. We also propose a way to convert existing simple dependency parser output to constituent*

Submission received: 24 May 2017; revised version received: 11 January 2018; accepted for publication: 1 March 2018.

doi:10.1162/COLI_a_00314

trees. This allows us to evaluate and to compare approaches from both constituent-based and dependency-based perspectives in a unified framework, using constituency and dependency metrics.

We thus propose an evaluation framework to compare extant approaches easily and uniformly, something the RST parsing community has lacked up to now. We can also compare parsers' predictions to each other across frameworks. This allows us to characterize families of parsing strategies across the different frameworks, in particular with respect to the notion of headedness. Our experiments provide evidence for the conceptual similarities between dependency parsers and shift-reduce constituency parsers, and confirm that dependency parsing constitutes a viable approach to RST discourse parsing.

1. Introduction

Discourse analysis takes various forms in the NLP literature, from mostly local relations between propositions in the Penn Discourse Treebank (Prasad et al. 2008) to structures covering a whole document, for instance, constituent structures (trees) in Rhetorical Structure Theory (RST) (Mann and Thompson 1988), or graphs between discourse elements in Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides 2003) or Graphbank (Wolf and Gibson 2006). In this article, we re-evaluate recent efforts to predict full discourse structures at the document level within a constituency-based approach to discourse analysis. We argue that the problem is more naturally formulated as the prediction of a dependency structure, on which simpler parsing strategies can be competitively applied. We show how to evaluate both families of discourse parsers in a constituent-based and a dependency-based framework.

Discourse aspects are becoming more and more present in various NLP tasks. Text-level structures are useful as a representation of the coherence of a text and its topical organization, with applications to summarization (Marcu 2000; Louis, Joshi, and Nenkova 2010; Hirao et al. 2013), sentiment analysis (Bhatia, Ji, and Eisenstein 2015), or document classification (Ji and Smith 2017). Most empirical approaches to this problem focus on predicting structures following RST and are based on the corresponding RST-Discourse Treebank corpus (RST-DT), a large annotated resource of texts for discourse analysis in English (Hernault et al., 2010; Feng and Hirst 2014; Ji and Eisenstein 2014; Joty, Cartenini, and Ng 2015).

On the other hand, there is a need to examine in depth some of the representational choices made within the RST framework. Many discourse parsers for RST have made simplifying assumptions with respect to the linguistic annotations for practical purposes, and these choices affect the generality of the models and their evaluation. We thus focus on the issue of predicting a structure for a text, comparing different representations over the RST-DT. We will analyze the impact of the practical choices one makes while doing discourse parsing, while taking into account specificities of the discourse annotations provided in the corpus.

RST is a constituency-based theory: Discourse units combine with discourse relations to form recursively larger units up to a global document unit. The linguistic descriptions and pictorial representations of these structures implicate a similarity to the constituents of phrase structure grammars. However, this similarity is more apparent than real; discourse structure even in its RST format is a relational structure, formalized as a set of "discourse constituents" together with a set of relations, instances of which hold between the constituents.

All of this suggests a different approach to discourse parsing—namely, "dependency parsing," which has gained some currency in the discourse parsing community (Muller et al. 2012; Hirao et al. 2013; Li et al. 2014). Discourse dependency parsing is analogous to syntactic dependency analysis, where units are directly related to each other without intermediate upper-level structures, to yield a directed tree structure over the discourse units of a text. Simple dependency structures are simplified representations of more complex SDRT graph structures (Muller et al. 2012), or a simplified representation of RST structures (Li et al. 2014), with some applications to specific tasks (Hirao et al. 2013). As we will show, this engenders some information loss. Once one takes the order of attachment into account, however, dependency representations do not in principle imply a loss of information, as established in the syntactic domain by Fernández-González and Martins (2015), using the notion of head-ordered dependency structures. Applying the head-ordered idea to discourse provides some advantages over the use of constituent structures, as head-ordered structures do not assume a higher abstraction level, a potentially contentious issue in discourse analysis (Hobbs et al. 1987). They also lend themselves to different computational models, as is also the case in syntactic parsing. We also relate this to how some work already makes use of a related notion of head constituent to inform discourse parsing.

Dependency structures are also more general, and allow for the different structures advocated by the various aforementioned theories: tree structures (projective or not) or graph structures. They can thus provide a common representation framework between the different existing corpora.

Dependency parsing also comes with different evaluation measures from those used in constituent-based parsing. We will discuss in detail the relationships between evaluation measures of these two approaches. Our discussion parallels those that emerged in the syntactic community about evaluations on constituents or grammatical relations (Carroll, Briscoe, and Sanfilippo 1998), even though discourse raises specific questions, which should benefit the discourse parsing community.

Our article is organized as follows: In Section 2 we present the traditional view of RST structures as constituency trees and point out specificities of the RST-DT treebank. Then in Section 3 we discuss the duality of discourse structures between constituency- and dependency-based representations, and present translation procedures from one to the other.

In Section 4, we present an extensive evaluation of existing RST discourse parsers on the test section of the RST-DT, using both constituency and dependency metrics, thus providing a uniform framework for comparing extant approaches to discourse parsing. We also analyze empirically how diverse parsing strategies are related to each other. Section 5 concludes.

2. Constituent-Based Representations of RST Discourse Structures

Although several theories of discourse structure have been proposed for both text and dialogue, discourse parsing work has largely concentrated on the RST-DT (Carlson, Marcu, and Okurowski 2003). The annotations of the RST-DT follow the general framework of RST (Mann and Thompson 1988).

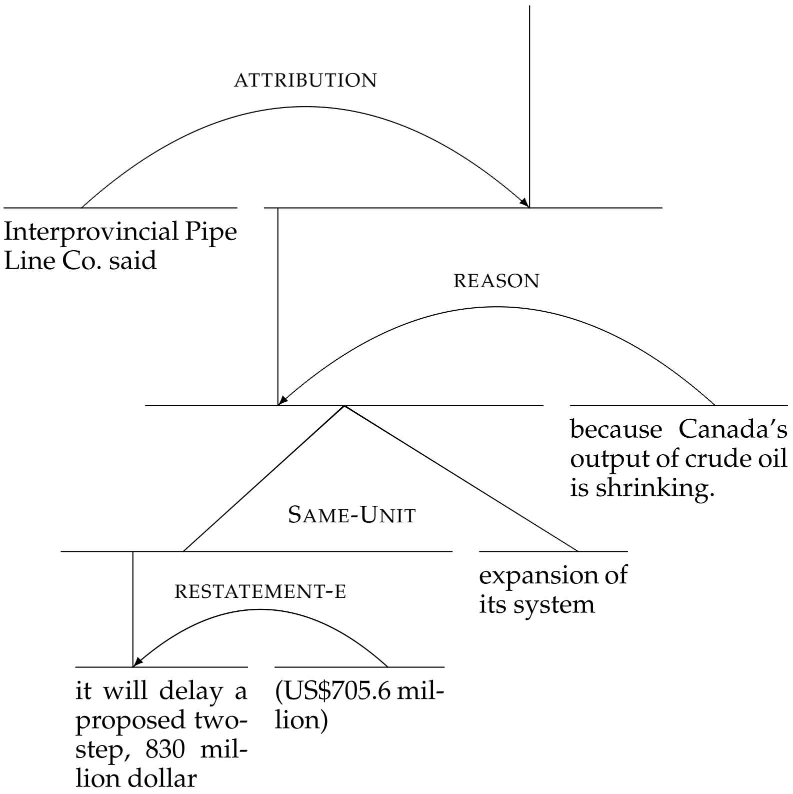

ATTRIBUTION
Interprovincial Pipe Line Co. said
REASON
because Canada's output of crude oil is shrinking.
SAME-UNIT
expansion of its system
RESTATEMENT-E
it will delay a proposed two-step, 830 million dollar
(US$705.6 million)

Figure 1
Example (1) in the RST-DT variant of the RST framework. In accordance to RST conventions, arrows point to the nucleus of the relation.

2.1 Original Representation of RST Structures

The traditional representation of discourse structure in RST is in the form of a tree with an apparent constituent structure. Let us consider an example[1] whose structure is illustrated in Figure 1 in the original presentation form.

(1) [Interprovincial Pipe Line Co. said]π_1 [it will delay a proposed two-step, 830 million dollar]π_2 [(US$705.6 million)]$\pi_3$ [expansion of its system]π_4 [because Canada's output of crude oil is shrinking.]π_5 (wsj$_{2363}$)

Brackets indicate what are considered as **elementary discourse units** (EDUs), which represent for the most part clauses. The segmentation model of the RST-DT only allows for continuous EDUs, hence intervening embedded clauses (parenthetical, appositive, or non-restrictive relative clauses), effectively splitting what semantically is a unique EDU into two or more fragments. In our example, π_2 and π_4 are two fragments of EDU split by the parenthetical π_3.

RST distinguishes between two types of relations on discourse units, depending on the relative importance of its members for the understanding of the text: **mononuclear**

[1] From the training section of the RST-DT.

relations link a unit of lesser importance, the **satellite**, to a unit of greater importance, the **nucleus**, and **multinuclear** relations link two or more units of similar importance, all nuclei. Relations constrain the recursive grouping of discourse units, from EDUs to a discourse unit spanning the entire text. A discourse unit is induced by either one multinuclear relation, or one or more mononuclear relations with the same nucleus; the subconstituents of a discourse unit are the members of its inducing relations. As a result, the discourse structure of a text in RST is a constituent tree of discourse units linked by relations. In the RST-DT, annotations are additionally constrained by the **adjacency principle**, whereby only neighboring units can be related. This implies that annotations in the RST-DT are continuous constituent trees.

In Figure 1, horizontal lines represent spans, discourse units that can be interpreted as being or as containing the arguments of discourse relations; labeled directed edges correspond to mononuclear relations where the source is the satellite and the target is the nucleus; vertical and diagonal lines link discourse units to their nuclei members. The name of each relation is indicated by a label attached either to a directed edge for mononuclear relations, like ATTRIBUTION, or a set of vertical and diagonal lines originating from the same horizontal line for multinuclear relations, like SAME-UNIT.

The inventory and granularity of discourse relations varies in different views of RST. The original presentation of RST uses 24 relations (Mann and Thompson 1988), whereas the RST-DT annotation guide (Carlson, Marcu, and Okurowski 2003) lists 76 "genuine" relations (53 mononuclear, 23 multinuclear), like ATTRIBUTION, REASON, and RESTATEMENT in Figure 1, plus two multinuclear pseudo-relations, TEXTUAL-ORGANIZATION and SAME-UNIT, used to "patch up" the tree structure. SAME-UNIT compensates for the fact that RST does not allow embedded EDUs. It relates fragments of an EDU separated by an intervening parenthetical clause, appositive clause, or non-restrictive relative clause. Intuitively in Example (1), "it will delay a proposed two-step, 830 million dollar [...] expansion of its system" to form a unique discourse unit split by the parenthetical "(US$705.6 million)"; their semantics is incomplete and needs to be combined. TEXTUAL-ORGANIZATION, not represented in the example, encodes conventional relations between distinct blocks of text in the same document, like that between a title, a body of text, a date, and a signature. RST uses this pseudo-relation for all conventional textual layouts, even though in some cases one could construct relations that specified the semantic and rhetorical function of these components. The 76 genuine relations are partitioned into 16 classes and each pseudo-relation has its own class, yielding the 18 classes of relations commonly used in RST discourse parsing. In the RST-DT corpus itself, some of the 53 mononuclear relations are further specialized to indicate that the relation is embedded, like RESTATEMENT-E in Figure 1, yielding a total of 112 labels.

2.2 The Semantics of RST Trees and the Reality of Constituents

Although such structures are familiar to most researchers on discourse structure, a rigorous, semantic interpretation of them was never part of RST, and is rarely discussed in computational linguistics.

The first thing to notice is a possible ambiguity in what might be the terms of a discourse relation; considering Example (1), the ATTRIBUTION relation might hold between the discourse constituent/span π_1 on the left *Interprovincial Pipe Line Co. said* and the span consisting of the following four segmented units ($\pi_2, \pi_3, \pi_4, \pi_5$) or some subset of these. In the example at hand, it is obvious from the context that

Interprovincial Pipe Line Co. said that it will delay the expansion of its system, and it is also quite probable that what they said did not include the content in which 830 million Canadian dollars are specified in U.S. dollar amounts. Concerning the last discourse unit π_5 *because Canada's output of crude oil is shrinking*, it's unclear whether this was part of what Interprovincial Pipe Line Co. said or not.

In RST, we can only partially represent this ambiguity. We can get at the "core" of the potential second argument of the ATTRIBUTION relation by making it the value of iteratively seeking the nuclei of a span until one comes to basic spans that have no discourse structure beneath them (EDUs). In our example, this idea, which is formalized under the heading of **the Nuclearity Principle** (Marcu 1996), would net us only the units (π_2, π_4)—corresponding to a fragmented EDU—*it will delay a proposed two-step, 830 million dollar [...] expansion of its system*. These "core" EDUs (π_2, π_4) constitute the **promotion set** of the discourse unit $[\pi_2 \ldots \pi_5]$ (Marcu 2000). On the other hand, one might choose not to use the Nuclearity Principle and accordingly take the entire span to the right $[\pi_2 \ldots \pi_5]$ as the argument of the ATTRIBUTION relation.

There is no mechanism in RST, however, that would yield as the second argument of the ATTRIBUTION relation the content given by EDUs (π_2, π_4, π_5) in the current annotation. One could make the REASON relation that holds between $[\pi_2, \ldots, \pi_4]$ and π_5 multinuclear, and one could have a version of the Nuclearity Principle which gathered then (π_2, π_4, π_5) into a **complex discourse unit** (CDU) that was the argument of the EXPLANATION relation. But this CDU would not correspond to a well-formed RST tree, because it does not correspond to a contiguous span. Although such structures are studied in other frameworks like SDRT (Venant et al. 2013; Perret et al. 2016), they have not been studied computationally nor even theoretically within the RST framework as far as we know. This simple example already points to a potential empirical problem with RST's conception of structure.

The relative unclarity of how to determine the arguments of discourse relations in an RST tree complicates efforts to capture semantically relevant information in these structures, and thus undermines a semantic argument for analyzing discourse in terms of constituent structures like RST tree structures. On the other hand, although most computational approaches have eschewed a principled answer to this question, almost all proposals agree that the EDUs should play a part in the relational structure, as should the relations that link them. But we need some sort of Nuclearity Principle in addition to RST trees to get at the relations between EDUs that are intuitively present. Only Venant et al. (2013), to our knowledge, in NLP circles has argued in favor of multiple EDUs or yet again substructures of the complete graph to be arguments of discourse relations, though Asher (1993) and Asher and Lascarides (2003) argue for this claim on linguistic grounds. But once again, as we have seen, these do not necessarily correspond to RST tree constituents.

If we compare complex constituents of syntactic structures and RST trees, syntactic constituents typically follow some sort of projection hypothesis and have a maximal type of structure for each basic category—namely, NP for N(ouns), DP for D(eterminers), VP for V(erbs), IP for I(inflection), and so on. And the nodes labeled with a basic category are the *heads* of the more complex constituent; thus, a node labeled *N* is the head of the complex constituent *NP*, and so on. Psycholinguists and linguists have adduced considerable evidence for the linguistic and cognitive reality of such projections by looking at phenomena like movement and extraction. But no such movement or extraction data exists for discourse structures, and in particular for RST trees or their complex constituents. Moreover, RST theory does not make non-elementary RST trees the projections of any head. As we will see, however, the notion of a constituent's head is crucial for RST parsing, and will be central for showing equivalences between discourse constituency-based representations (henceforth c-trees) and dependency-based representations (d-trees).

2.3 Specificities of RST Constituent Trees

We present here a few important aspects of practical efforts to deal with RST structures, going beyond the theoretical model.

Labeled tree structure encoding. In the RST-DT corpus, RST trees are stored as constituent trees, where each constituent has a label that can contain a relation name and nuclearity status. The satellite of a mononuclear relation carries the name of the relation and has nuclearity *Satellite*, while the nucleus of a mononuclear relation is labeled SPAN and *Nucleus*. All nuclei of a mononuclear relation are labeled with the name of the relation and have nuclearity *Nucleus*. The root node does not carry any relation name; we denote it with the empty word ϵ, and assign it nuclearity *Root*; The RST tree from Figure 1 can be alternatively represented as the "pure" constituent tree in Figure 2, where nuclearity statuses are represented by their initial letter (N, S, and R for Nucleus, Satellite, and Root).

Marcu (2000) introduced this encoding of RST trees to represent RST trees of arbitrary arity, including discourse units consisting of a nucleus independently modified by two satellites via different relations, in a format suitable for training and evaluating parsers. Without these specific constructs, however, a more natural encoding would have put the relations labels higher up, on the node dominating the related constituents. This raises problems we will come back to in Section 2.4. The standard evaluation procedure for RST discourse parsers is Marcu's (2000) adaptation of the Parseval procedure to this encoding, which we name **RST-Parseval**.

Binarization. It is standard practice since the first proposals on RST discourse parsing (Reitter 2003; Soricut and Marcu 2003) to train and evaluate parsers on a right-heavy

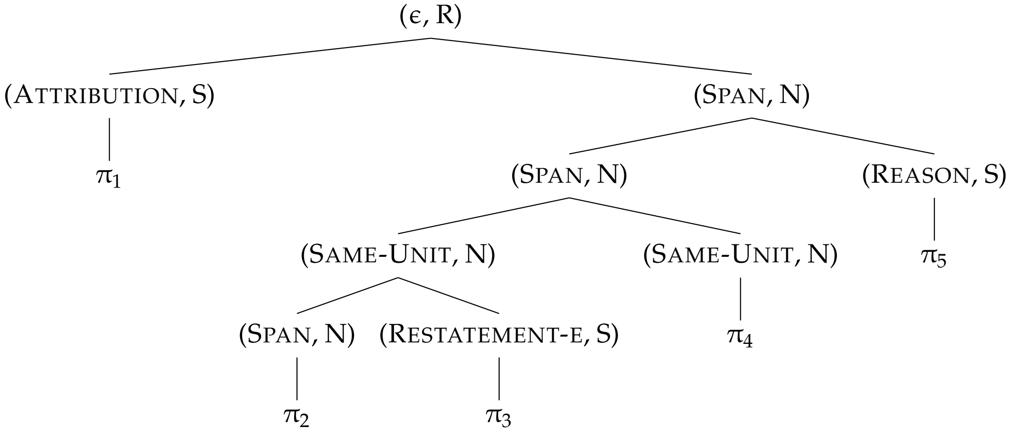


Figure 2
Constituent tree corresponding to the RST tree in Figure 1.

binarized version of the c-trees from the RST-DT. This is mostly accidental: constituency-based RST discourse parsers inherit a limitation shared by most syntactic constituency-based parsers that operate better, if not only, on binary trees.

Such binarization is, however, not a theoretically innocent choice, because it can introduce new intermediary nodes that change the semantic interpretation of the trees (van der Vliet and Redeker 2011). The structural changes induced in the tree structure and the number of nodes also affect the evaluation metrics that are used to assess the performance of discourse parsers, as we will see in Section 4. Even though *n*-ary trees are not so frequent in the RST-DT corpus, binarization applied high up in the original tree can drastically change an evaluation score. Figure 8 later in this article shows what happens when binarized structures differ (left vs. right): all relevant arguments' substructures have different spans, generating multiple errors down the tree. Theoretically and empirically, the binarization of RST c-trees is not trivially reversible, a point that is overlooked in most studies except Sagae (2009) and Hernault et al. (2010).

Headed RST c-trees. In the formal presentation of RST trees by Marcu (2000), each node is assigned a **promotion set** containing its most important EDUs, in accordance with the Nuclearity Principle. The promotion set of a node is recursively defined as the union of the promotion sets of its nuclei, the promotion set of each leaf (EDU) being the EDU itself. Semantically, promotion sets are said to provide a validation criterion for relations between wider discourse units: A relation can hold between two discourse units if it also holds between their promotion sets. This leads Marcu to use the promotion set of (non-elementary) discourse units as a proxy to extract semantic similarity (WordNet)-based features between these units, in his early shift-reduce RST discourse parser (Marcu 2000).

Sagae pursued Marcu's line of work by adapting the shift-reduce parser he had developed for syntactic parsing (Sagae and Lavie 2005) to RST discourse parsing (Sagae 2009). Sagae's syntactic parser is a lexicalized shift-reduce parser that simultaneously produces both a dependency and a constituent structure, using features related to either of the two types of structures being built. Building an RST tree from EDUs is essentially similar to building a syntactic tree from tokens, save for the fact that discourse units have a promotion *set* rather than a unique *head*, because discourse relations can relate two or more nuclei of equal importance whereas syntactic dependencies (usually) asymmetrically relate a head and a dependent. To adapt his parser to RST parsing, Sagae therefore defined a variant of RST trees where each discourse unit has a head EDU in its span, recursively defined as the head EDU of its leftmost nucleus. The notion of head EDU is instrumental to the formal equivalence between c-trees and d-trees we show in Section 3.3. Its importance in parser design explains the structural similarity between the predictions of constituency shift-reduce parsers, for example (Ji and Eisenstein 2014; Braud, Coavoux, and Søgaard 2017), and dependency parsers, as we will see in Section 4. However, two different sequences of (shift or reduce) actions can produce the same set of dependencies but different constituent trees, reflecting different orders of attachment of dependents to their heads; hence to fully reflect constituent structures, a dependency structure must incorporate a notion of order, as we will see in Section 3.3.

2.4 Evaluation Procedures

We present here the different evaluation measures used in the discourse parsing literature, focusing on constituent-based approaches. We will see that even within that framework there are variants, which can lead to potential confusions when comparing different approaches. We will consider dependency-based evaluation in Sections 3.5 and 3.6, and see how they differ in the kind of information they measure. The differences partially reflect similar discussions within the syntactic community, where dependency structures have been advocated in part because of issues with constituent-based evaluation (Carroll, Briscoe, and Sanfilippo 1998; Carroll, Minnen, and Briscoe 1999). Constituency-based approaches naturally lend themselves to evaluation procedures that originated in the syntactic parsing literature.

The most common evaluation procedure is Parseval (Black et al. 1991), which combines precision and recall on (typed) constituents, plus a Crossing Parentheses score, which is the ratio of the number of predicted constituents that cross a gold standard constituent with respect to the total number of predicted constituents. Early work on RST parsing (Marcu 2000) introduced a modified version of Parseval to accommodate RST c-trees encoded as outlined above. This early work addressed the two tasks of segmenting elementary discourse units and parsing. As almost all existing work assumes gold segmentation and focuses on the parsing task itself, and because the trees are binarized, there are a few simplifications to consider. First, precision and recall are equal if admissible structures are binary trees, and thus existing work only provides accuracy scores. Note that this is not necessarily the case for n-ary structures, since the number of predicted and reference spans can then be different.

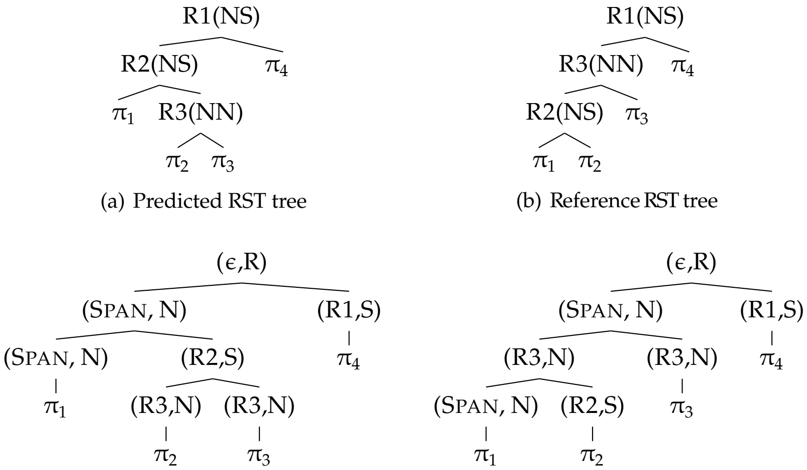
On the contrary, a binary tree spanning n EDUs is always made of $n-1$ binary constituents, resulting from $n-1$ attachment decisions, thus containing a total of $n+(n-1)=2n-1$ "spans."

Counting crossing brackets is not necessary because this number is directly correlated to the accuracy on the spans, whereas in syntactic parsing this measure is necessary to mitigate the effect of missing spans (empty categories, for instance) on pure precision and recall.

RST parsing studies usually consider the unlabeled spans (S) as the basic building blocks to evaluate, irrespective of the discourse relation that links two discourse units to form a constituent. The discursive function of correctly identified spans is then evaluated, either on the sole label of the incoming relation (R) or on the assignments of nuclearities to the arguments (N) or both (F). We note also that all of these evaluations provide a kind of edit distance over RST c-trees, and are hence true metrics (as is any linear combination of them).

Nuclearity in RST is a type of information that can provide a partial notion of directionality, as Figure 1 shows, for the case of mononuclear relations. For multinuclear relations there is no directionality, as the two arguments are of "equal importance." Constituency-based RST parsers should distinguish three types of nuclearity in their binary attachment decisions: "NS" if the left subtree is the nucleus of a mononuclear relation and the right subtree is the satellite, "SN" if the linear ordering of the satellite and nucleus is inverted, and "NN" for multinuclear relations.

Figure 3 shows an example of a scoring of a constituency tree against a reference, with all four different scores (S), (N), (R), (F). A consequence of assuming the gold segmentation for evaluation of existing discourse parsing methods is that accuracy on the spans should be restricted to non-EDUs. Otherwise, the span measures will be overestimated, since the n basic spans (EDUs) are necessarily correct, and only $(n-1)$ spans corresponding to binary relations on top of the EDUs are predicted; for instance, a predicted RST tree whose structure is entirely different from the reference would still have a full unlabeled span (S) score of $n/(2n-1)$. Excluding the EDU spans from the evaluation is not the general practice in the literature, however, because as we have



(a) Predicted RST tree

(b) Reference RST tree

(c) Predicted, RST-DT format

(d) Reference, RST-DT format

	Predicted Spans	Correct	Nuclearity	Correct	Relation	Correct
Original RST	π_2, π_3	×	NN	×	-	×
	π_1, π_3	✓	NS	×	R2	×
	π_1, π_4	✓	NS	✓	R1	✓
	Scores	S = 2/3		N = 1/3	R =	F = 1/3
RST-DT	π_1	✓	N	✓	SPAN	✓
	π_2	✓	N	×	R3	×
	π_3	✓	N	✓	R3	✓
	π_4	✓	S	✓	R1	✓
	π_2, π_3	×	S	×	R2	×
	π_1, π_3	✓	N	✓	SPAN	✓
	π_1, π_4	✓	R	✓	ϵ	✓
	Scores	S = 6/7		N = 5/7	R =	F = 5/7

(e) Scores

Figure 3
Example of constituent-based evaluation of a predicted tree in the "original" RST format (a) with respect to a reference (b), with four scores: span (S), span+nuclearity (N), span+relation type (R), span+nuclearity+relation (F). This evaluation ignores EDU spans, which are assumed to be given. In the RST-DT format, the predicted and reference trees are now (c) and (d), with additional artificial relations SPAN and inflated scores. Note that multinuclear relations are duplicated on all their children.

seen, in the format of the RST-DT treebank, mono-nuclear relations are stored onto their satellite child, and the nucleus is labeled SPAN (cf. Figure 3(c)-3(d)). In the case of multinuclear relations, all nuclei receive the relation label. It is thus easier to include the spans corresponding to EDUs into the evaluation procedure to recover their relation and nuclearity status.

In addition, the evaluation procedure followed by every cited paper considers SPAN relation labels as bona fide discourse relations, although all they do is (by construction) count again the given span as the nucleus, something that is already taken into account in the Nuclearity score. This adds another "pseudo-relation" to RST's inventory; a significant proportion of the evaluation of relation labels thus corresponds in fact to the correct identification of the nuclei of mononuclear relations. This problem is one of the arguments put forward by Iruskieta, da Cunha, and Taboada (2015) to propose an alternative evaluation metric for discourse trees that is actually similar to the dependency metrics we present in the next section. The influence of this issue on the scoring of labeled spans in the literature will be empirically investigated in Section 4.

3. A Dependency View of RST Trees

In this section, we investigate in depth the correspondence between dependency structures and RST trees. We first provide a brief overview of extant work on this topic. We then define a dependency structure that is fully equivalent to an RST constituent tree using the standard convention proposed by Sagae (2009) for determining heads in RST constituent trees. Converting RST trees to dependency structures requires us to add some structure to the canonical definition of an RST tree, in particular the notion of a head. The next step is to introduce a dependency structure with nuclearity information (RST dependency trees). We will then exploit **head-ordered dependency trees**, a formalism proposed in syntactic parsing (Fernández-González and Martins 2015) to deal with ambiguities of scope that arise when translating from constituent structures to dependency structures. We will show that head-ordered dependency structures encode all the information in a headed RST tree, so that there is a bijection between RST trees and head-ordered, projective dependency trees. We also consider the aspects related to the evaluation of parsers producing dependency structures.

In the following, we will sometimes abbreviate constituent structures and trees as c-structures and c-trees, and similarly dependency structures and trees as d-structures and d-trees.

3.1 An Introduction to Dependency Structures for RST

The duality of representation structures found in syntax between constituents and dependencies is now part of the domain of discourse parsing, too. The target representations are similar: Elementary units (words or discourse units) combine to form larger structures, either by grouping (constituents) or by linking (dependencies). As we have seen, the RST Discourse Treebank is an example of the first kind, with specificities presented in the previous sections: Only adjacent units can be grouped together, and all units must be grouped into one that spans the entire document, in a tree-like structure. Although dependency structures have been advocated for discourse in general in Muller et al. (2012) and applied to an SDRT corpus, they have been applied specifically to RST annotations in Hirao et al. (2013), and predicted as the output of an RST parser in Li et al. (2014) and Yoshida et al. (2014).

Basic dependency trees for discourse are of the form: $D = \{(n_1, n_2, rel) \colon n_i \in E, \text{ and } rel \in R\}$, where D is a directed spanning tree over a set of EDUs E with labels from a set of discourse relation labels R on the edges. Directedness implies that each dependency tree has a unique head or governor and one or more dependents, and this means the concept of head/governor must be defined for discourse.

We have seen in the previous section that the non-atomic constituents of RST constituent trees are not straightforward to interpret semantically. Dependency trees have a semantic transparency that is lacking with the constituent structures of RST. If CDUs are not intended as arguments of relations, then the natural, compositional interpretation of a dependency tree gives us the actual semantic arguments of each discourse relation. If CDUs are intended as arguments of relations, then a dependency structure tells us at least one of the CDU's constituents and hence predicts a part of the argument of the relation. The RST constituent structure on its own does neither of these things, though one could interpret the constituent structure without the nuclearity principle as providing another sort of underspecified semantic structure: If a relation has an RST subtree as an argument, then the "real" semantic argument of the discourse relation is to be found somewhere in that subtree or is in fact the whole subtree.

Another virtue of a dependency-based framework is that it generalizes naturally from RST constituent trees to other more permissive discourse structures. Frameworks like SDRT or GraphBank assume more general structures than trees (respectively, directed acyclic graphs and arbitrary graphs). Venant et al. (2013) show that differences between frameworks can be encoded with different, additional structural constraints on a dependency based graph. Thus, dependency graphs capture commonalities between these different types of representation, and it is easy to convert them into a dependency graph between EDUs. It was already shown in Danlos (2004) that some discourse constructs would be more appropriately represented as directed acyclic graphs.

Although we will show below how dependency parsing can provide comparable analyses to constituent-based parsers for the RST corpus, an added advantage of dependency parsing formalisms like the one we have proposed here is that it is easily adaptable: (1) to discourse structures that are not projective trees—Muller et al. (2012) use non-projective maximum spanning tree (MST) algorithms to compute dependency trees in an SDRT style on the Annodis corpus of French texts; (2) to structures that are not even treelike, for instance, representations for multi-party chat conversations—Perret et al. (2016) use the STAC corpus (Asher et al. 2016) where discourse annotations are not trees but only directed acyclic graphs, which constituent parsers based on RST projective tree formalisms cannot reproduce. Perret et al. (2016) have shown how a dependency formalism combined with constraints implemented in ILP can learn such structures with comparable measures of success to the efforts on the more constrained RST tree bank. Dependency parsing also easily handles long distance dependencies without the computation of nuclearity features. Venant et al. (2013) have shown that nuclearity computations cannot handle all long distance dependencies and argue that it is difficult to give a sensible semantic interpretation of structures that use nuclearity features to compute long distance dependencies. Finally, the STAC and Annodis corpora contain discourse structures in which substructures of the whole graph can act as discourse constituents. These are known as *complex discourse units*. While dependency graphs on their own do not suffice to fully represent CDUs, Perret et al. (2016) also shows how dependency structures can approximate CDUs by distributing the relations in which a CDU is an argument over the CDU's member EDUs.

Dependency structures for discourse formalisms less constrained than RST's constituent trees are easy to make and to use, because dependency structures are semantically more transparent than c-structures. The same semantic interpretation relation holds when we relax dependency trees to non-projective trees as in Muller et al. (2012) and Afantenos et al. (2015) or to directed acyclic graphs as in Perret et al. (2016). CDUs aside, dependency structures exhibit the actual semantic arguments of the relations and are much closer to a logical formalism in which binary relations could hold in all sorts of graph configurations. A constituency based approach like RST's arguably does not exhibit the actual semantic arguments of discourse relations—hence, the need for a nuclearity principle.

3.2 Dependency-Based Alternatives to RST Trees

As we have just seen, the notion of a head is central to a dependency structure for a discourse. The notion of headedness is also crucial for both a semantically transparent and structurally sound notion of discourse tree structure. All discourse relations that have been proposed that have any nontrivial semantic content have an implicit order; an explanation relation distinguishes an *explanans* and an *explanandum*; a causal relation like Result distinguishes between a cause and effect; a thematic relation like Elaboration distinguishes between a theme or topic element and an element that develops the theme. Headedness captures a crucial aspect of relational structures, and if discourse structures are anything they are that. We need a notion of headedness to tell us which way the semantics of the relation is to be directed. We also need a notion of headedness to determine exactly what sort of tree structure is proposed. That is why we propose to move from traditional c-trees to headed c-trees.

Although RST trees lack a notion of headedness, we can exploit the nuclearity decorations of RST trees together with a heuristic to produce headed trees and eventually dependency trees from classic RST trees. It is important to emphasize, however, that nuclearity decorations that apply to *spans* do not determine in and of themselves a head EDU for a tree. When these spans are non-atomic (i.e., they contain more than one EDU), nuclearity tells us little about the actual arguments of the eventual relation that links these spans. In any case, nuclearity by itself tells us nothing about the direction of the eventual relation, unless we add an interpretive principle of nuclearity that tells us how to use this feature to calculate the headedness of a structure or direction of an attachment. As we have been at some pains to point out, nuclearity and headedness are not the same thing; headedness does not determine nuclearity, because where a and b are EDUs, the two simple trees $[a_N, b_S]$ and $[a_N, b_N]$ (since we are just looking at structure and nuclearity, we have omitted any of the relational decorations) have the same head. And nuclearity by itself does not determine headedness even though its introduction by Marcu (2000) was semantically motivated in an effort to find the core or central idea in a span; a notion is reflected in other theories (Grosz and Sidner 1986; Asher 1993) by the use of coordinating and subordinating discourse relations. Consider, for instance, the following hypothetical tree structure $[[a_N b_N]_N, c_S]$, where $[a_N b_N]$ is a subtree and the nucleus of the larger tree that includes c. An admittedly somewhat crazy interpretive principle that takes the head of the multinuclear relation to be the rightmost one locally but takes the leftmost one as the head projected up through the tree would yield the following non-projective dependency structure: $\{a \leftarrow b, a \rightarrow c\}$. A saner, interpretive principle would be the one that we suggested earlier, producing a standard projective tree: $\{a \rightarrow b, a \rightarrow c\}$. The point is, the nuclearity measure by itself is not telling us much about the structure of the tree that is relevant to the semantics.

Thus, we need a heuristic to go from nuclearity decorations on spans to head EDUs. Sagae (2009) was the first to propose such a heuristic to define heads on RST trees, motivated by technical aspects of his shift-reduce parsing algorithm. Others, like Ji and Eisenstein (2014) and Braud, Coavoux, and Søgaard (2017), have adopted the same parsing algorithm and inherited the same notion of headedness. The heuristic for determining a head EDU from nuclearity decorations is simple in the case of trees with only mononuclear relations. However, if the structure contains a subtree with a multinuclear relation, then a decision that is arbitrary from the point of view of the RST constituent structure must be made as to what is the head of the subtree; one can either take the left or the right argument in an NN structure to be the head. Without a notion of head, two dependency trees will represent one RST tree consisting of one multinuclear relation. On the other hand, assuming some convention to determine the head in an RST tree will result in a dependency structure that represents two RST c-trees; for instance, assuming that a tree consisting of one multinuclear relation has its head in the leftmost argument will mean that the RST trees $[a_N, b]$ and $[a_N, b_N]$ yield the same unlabeled dependency structure, $a \to b$. Consequently, RST non-terminal constituents have been translated as dependencies with some information loss; see for example (Li et al. 2014). We will return to this problem in subsection 3.3; see Figure 5.

Inspired by similar proposals in syntax and in other discourse formalisms relying on head percolation, two variants of dependency trees were independently proposed as an explicit alternative to RSTc-trees (Hayashi, Hirao, and Nagata 2016). In the first variant (Li et al. 2014), each non-terminal is assigned a head EDU, which is the head EDU of its leftmost nucleus child, following the same heuristic as Sagae (2009). Then, a dependency relation is created for each non-terminal from its head to its dependent, in a procedure similar to those designed for syntax. An example RST c-tree and the corresponding d-tree are presented in Figure 4. The second variant (Hirao et al. 2013) differs from the first one in its processing of multinuclear relations: A multinuclear relation in the RST c-tree does not translate directly to a dependency relation but instead "distributes" the closest mono-nuclear relation higher in the tree (Yoshida et al. 2014; Hayashi, Hirao, and Nagata 2016). The difference between these two variants is explained by the different empirical motivations behind their development.

Li et al. (2014) adapt a syntactic dependency parser, the MSTParser of McDonald, Crammer, and Pereira (2005), to predict RST dependency trees over a text segmented in EDUs instead of syntactic dependency trees over sentences segmented in tokens. They convert the RST constituency trees from the RST-DT to RST dependency trees that they use for training and evaluation. They report experiments with two decoders implemented in the original MSTParser, the non-projective Maximum Spanning Tree decoder and the projective Eisner decoder (Eisner 1996). Although the authors initially point out that a given simple RST dependency tree can correspond to several RST constituency trees, they do not describe any mechanism to cope with this issue and yet report RST-Parseval scores for projective variants of their parser. It turns out that their implementation of the RST-Parseval procedure does not operate on pairs of reference and predicted RST constituency trees but rather on pairs of reference and predicted RST dependency trees. For each RST dependency tree, constituency spans are computed from the transitive closure of its adjacency matrix (i.e., its maximal projection). The

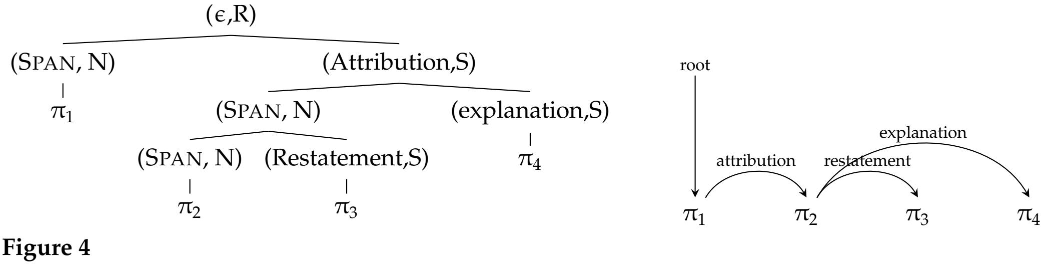


Figure 4
Example RST c-tree (left) and the corresponding d-tree (right). In accord with syntactic conventions, arrows point away from the head of the dependency.

same ambiguous configurations, however, arise in this formulation, which they solve by arbitrarily attaching to a head all the modifiers on its left, then all modifiers on its right. Their implementation of the RST-Parseval procedure thus manipulates reference spans that are absent from the original RST constituency trees. As a further consequence, the RST-Parseval scores reported by Li et al. (2014) are misleading because their implementation of the RST-Parseval procedure uses distorted reference trees, not the original annotations, which artificially raises scores by several points.

The second approach to RST dependency parsing started as a way of doing summarization, converting the output of a constituency-based RST parser to dependencies (Hirao et al. 2013). Yoshida et al. (2014) pushed this work further and used a dependency parser built around an MST decoder and trained on dependency trees converted from the RST-DT, using a classical training loss function and another one that puts more weight on dependencies higher up in the tree. In both settings, their parser attained better performance on the summarization task than a similar constituency-based parser. A subsequent study (Hayashi, Hirao, and Nagata 2016) refined these results in a comparative evaluation, both intrinsic and extrinsic, of three dependency schemes for RST structures, including the "tree" transform presented above and used in Li et al. (2014). The DEP-DT dependency scheme proposed by Hirao et al. (2013) was adopted by Bhatia, Ji, and Eisenstein (2015) to perform sentiment analysis.

As we will see formally in Section 3.3, although an RST constituency tree can be deterministically mapped to a simple RST dependency tree, the reverse does not hold true. A dependency tree can correspond to several constituency trees, depending on the relative order of attachment of modifiers located on both sides of a given head. This ambiguity hampers the direct evaluation of dependency-based RST parsers with the standard RST-Parseval procedure. Before seeing how to overcome this problem, we will present traditional evaluation procedures for evaluation structures. We will now see how to make a precise correspondence between dependency structures and RST constituents.

3.3 Headed RST Trees as c-trees

We start with a formal definition of the headed RST trees of Sagae (2009). Our definition of headed RST trees is essentially the same as the definition of the headed syntactic constituent trees in Fernández-González and Martins (2015), with minimal adaptations to account for the specificities of RST.

Let $e_1 e_2 \dots e_L$ be a document, where e_i denotes the EDU in the ith position. Let $\mathcal{R}$ be the set of relations and $\mathcal{N} = \{N, S, R\}$ be the set of possible nuclearities, short for *Nucleus*, *Satellite*, and *Root*, respectively.

Definition 1
A **headed RST tree** is a rooted tree whose leaves are the EDUs $\{e_i\}_{i=1}^{L}$, and whose internal nodes (constituents) are represented as a tuple $\langle Z, h, \mathcal{I} \rangle$, where:

1. $Z = \langle rel, nuc \rangle$ is a non-terminal tuple such that $rel \in \mathcal{R} \cup \{\text{SPAN}, \epsilon\}$ is either a relation, the label SPAN, or the empty symbol; and $nuc \in \mathcal{N}$ is the nuclearity of the node,
2. $h \in \{1, \dots, L\}$ indicates the head EDU,
3. $\mathcal{I} \subseteq \{1, \dots, L\}$ is the yield of the node.

ϵ and *Root* are reserved symbols for the root node of the RST tree: $Z_{root} = \langle \epsilon, Root \rangle$. The parent of each EDU e_i is a pre-terminal unary node of the form $\langle Z, i, \{i\} \rangle$. The yields and head EDUs are defined so that for every constituent $\langle Z, h, \mathcal{I} \rangle$ with children $\langle X_k, m_k, \mathcal{J}_k \rangle_{k=1}^{K}$, (i) we have $\mathcal{I} = \bigcup_{k=1}^{K} \mathcal{J}_k$; and (ii) there is a unique k such that $h = m_k$. This k-th node (called the head-child node) is commonly chosen as the leftmost nucleus child: $k = min(j \in \{1, \ldots, K\})$ such that $X_j = \langle rel_j, Nucleus \rangle$.

Non-terminal nodes in a headed RST tree obey the following constraint concerning nuclearity:

Constraint 1
For every constituent $\langle Z, h, \mathcal{I} \rangle$ with children $\langle X_k, m_k, \mathcal{J}_k \rangle_{k=1}^{K}$, either:

1. all but one child are satellites of the remaining child, each through a mononuclear relation: $\exists ! k. Z_k = \langle \textsc{Span}, N \rangle$ and $\forall k' \in \{1, \ldots, K\}. k' \neq k. \exists r' \in \mathcal{R}. Z_{k'} = \langle r', S \rangle$; or
2. all children are nuclei of a unique multinuclear relation: $\exists ! r \in \mathcal{R}. \forall k \in \{1, \ldots, K\}. Z_k = \langle r, N \rangle$.

Trees in the RST-DT follow the **adjacency principle**, whereby only neighboring units can be related. Formally, this means that RST trees are **continuous**, namely, all nodes $\langle Z, \mathcal{I} \rangle$ have a contiguous yield $\mathcal{I}$: $\forall i, j \in \mathcal{I}$ $(i < j \rightarrow \forall k (i < k < j \rightarrow k \in \mathcal{I}))$.

For what follows, we reproduce a few additional definitions from Fernández-González and Martins (2015) that apply unchanged to RST trees. An internal node that is not a pre-terminal is called a **proper node**. A node is called unary if it has exactly one child. An RST tree without unary proper nodes is called **unaryless**. If all proper nodes have exactly two children then it is called a **binary** tree. For each EDU position $h \in \{1, \ldots, L\}$, we define $\phi(h)$ as the highest node in the c-tree whose head EDU is h. We call the path from $\phi(h)$ down to the pre-terminal p_h the **spine** of h.

Definition 1 implies that, by construction, RST trees are unaryless. As already mentioned, existing RST discourse parsers additionally work on binary trees only, a requirement that facilitates learning and prediction in their parsing models: bottom–up greedy, CKY, or shift-reduce parsers.

An **RST dependency tree (d-tree)** is a single-rooted directed tree spanning all the EDUs in the document. Each arc in this tree is a tuple $\langle h, m, \ell \rangle$, expressing a typed discourse relation with nuclearity $\ell = \langle rel, nuc \rangle$ between the head EDU e_h and the modifier e_m.[2] A d-tree is **projective** if for every arc $\langle h, m, \ell \rangle$ there is a directed path from h to all EDUs that lie between h and m in the surface string.

Head-ordered RST Dependency Trees. Gaifman (1965) showed how to obtain projective dependency trees from headed continuous constituency trees[3] by reading off the head and dropping the internal nodes. However, in our case the relation between continuous c-trees and projective d-trees is many-to-one: as shown in Figure 5, several c-trees may project onto the same d-tree, differing on their flatness and on left or right-branching decisions.

[2] Note that nuclearity here is needed, because the direction of the arc in the dependency tree does not suffice to determine a nuclearity decoration, as we have discussed in Section 3.2.

[3] A continuous tree is one according to which the span of any subtree is an uninterrupted sequence of terminal constituents. This entails and is equivalent to a tree with no crossing dependencies.

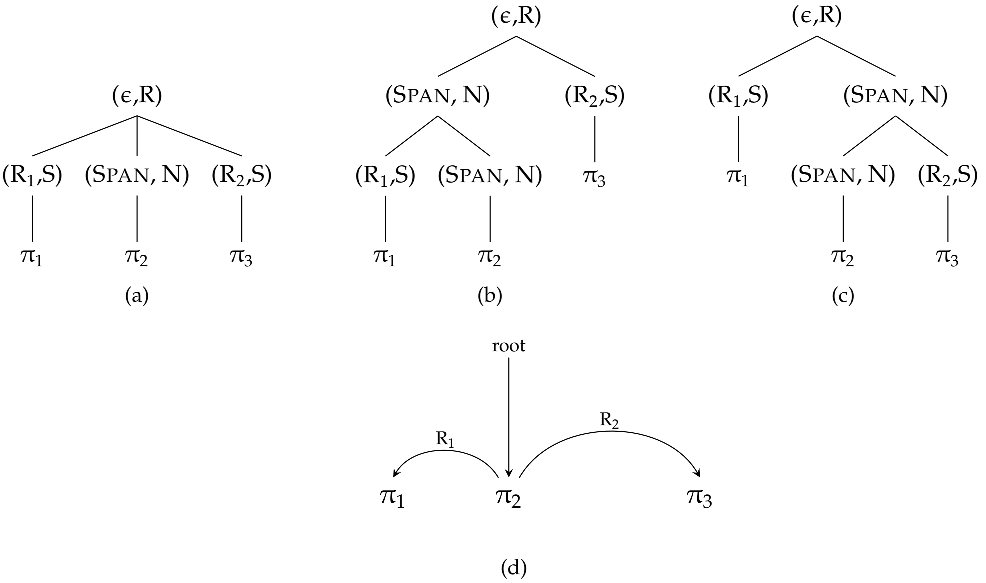


Figure 5
Ambiguity of the simple conversion of RST trees to dependency trees: three distinct RST trees (a)–(c) yielding the same dependency tree (d).

This ambiguity prevents a 1-1 mapping between c- and d-trees. To sum up, different RST constituent trees can yield the same dependency structure, since a head can be propagated through different constituent paths while being attached to the same satellites. We can, however, be more specific as to when this ambiguity arises. The adjacency principle in RST, whereby only adjacent spans can be attached, implies that dependents appearing on the same side of the governing unit are necessarily attached from the inside to the outside. It follows that ambiguities arise in specific configurations where a unit has dependents on both sides: the relative ordering of attachment of dependents on the same side is fixed, but it is compatible with several relative orderings of dependents from both sides. The source of ambiguity is the alternation between sides. We will call such ambiguous subtrees "spiders."

What's missing in our d-tree representation is the order according to which modifiers are attached to the head. Fernández-González and Martins (2015) solved this issue by adding to the d-tree a layer of information that stores the relative order of attachment of modifiers to each head. The resulting structure is called a **head-ordered dependency tree**. For each head EDU h with modifiers $M_h = \{m_1, \ldots, m_K\}$, we endow M_h with a **weak order relation** defined as an **ordered partition**: We can partition the K modifiers into J equivalence classes, $M_h = \bigcup_{j=1}^{J} \bar{M}_h^j$, and define a strict order $\prec_h$ on the quotient set: $\bar{M}_h^1 \prec_h \ldots \prec_h \bar{M}_h^J$. Intuitively, there is a sequence of events of attachment of modifiers (1 to J) to the head h, where at each event j one or more modifiers (the ones in the equivalence set $\bar{M}_h^j$) are simultaneously attached to h. We represent this graphically by decorating d-arcs with indices (#1, #2, . . .) to denote the order of events. A **weakly ordered RST d-tree** is an RST d-tree endowed with a weak order for each head.

Figure 6 shows one example RST tree from Figure 5 translated as a tree of this type.

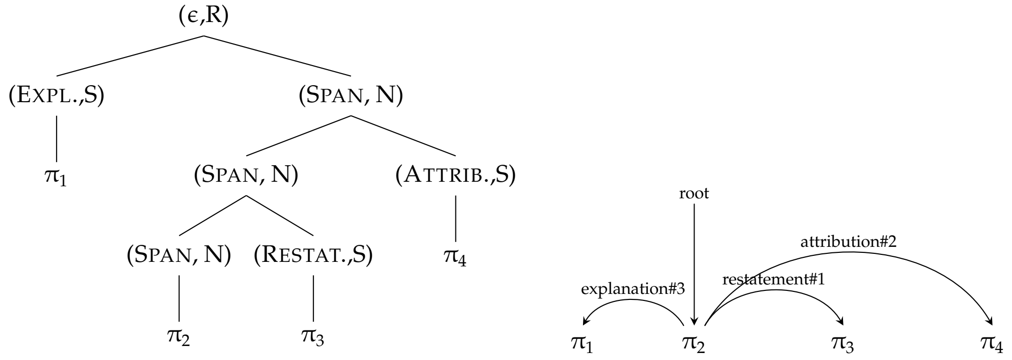


Figure 6
Correspondence between a head-ordered dependency tree and a headed RST tree.

3.4 Conversion from RST c-trees to RST d-trees

Having defined headed RST c-trees and ordered d-trees, we now show an isomorphism between them. We will need the following two algorithms from c-trees to d-trees and back.

We first detail a translation into dependency trees we have defined. The translations to dependencies in the already cited (Muller et al. 2012; Hirao et al. 2013; Li et al. 2014) all follow the general algorithm (Algorithm 1): each non-atomic constituent has a distinguished atomic constituent as its head, and this head is percolated recursively through larger constituents. We only add the numbering of attachment order, with a treatment of nodes given by a post-order traversal of the starting c-tree.

Algorithm 1 Conversion from RST c-tree to weakly-ordered RST d-tree

```
Input: RST c-tree $\mathcal{C}$
Output: weakly-ordered RST d-tree $\mathcal{D}$
  $\mathcal{D} := \{\}$
  for each $h = 1, \ldots, L$ do                    ▷ initialize order of attachment
      $j(h) := 1$
      $\bar{M}_h^{j(h)} := \{\}$
  end for
  Nodes := GetPostOrderTraversal($\mathcal{C}$)
  for each $v := \langle Z, h, \mathcal{I} \rangle \in$ Nodes do
      for each $u := \langle X, m, \mathcal{J} \rangle$ which is a child of $v$ do
          if $m \neq h$ then
              $e := \langle h, m, X \rangle$
              $\mathcal{D} := \mathcal{D} \cup e$
              $\bar{M}_h^{j(h)} := \bar{M}_h^{j(h)} \cup e$
          end if
      end for
      $j(h) := j(h) + 1$
      $\bar{M}_h^{j(h)} := \{\}$
  end for
```

To do the reverse transformation, we adapt the algorithm in Fernández-González and Martins (2015) to binary RST trees, as in Algorithm 2.

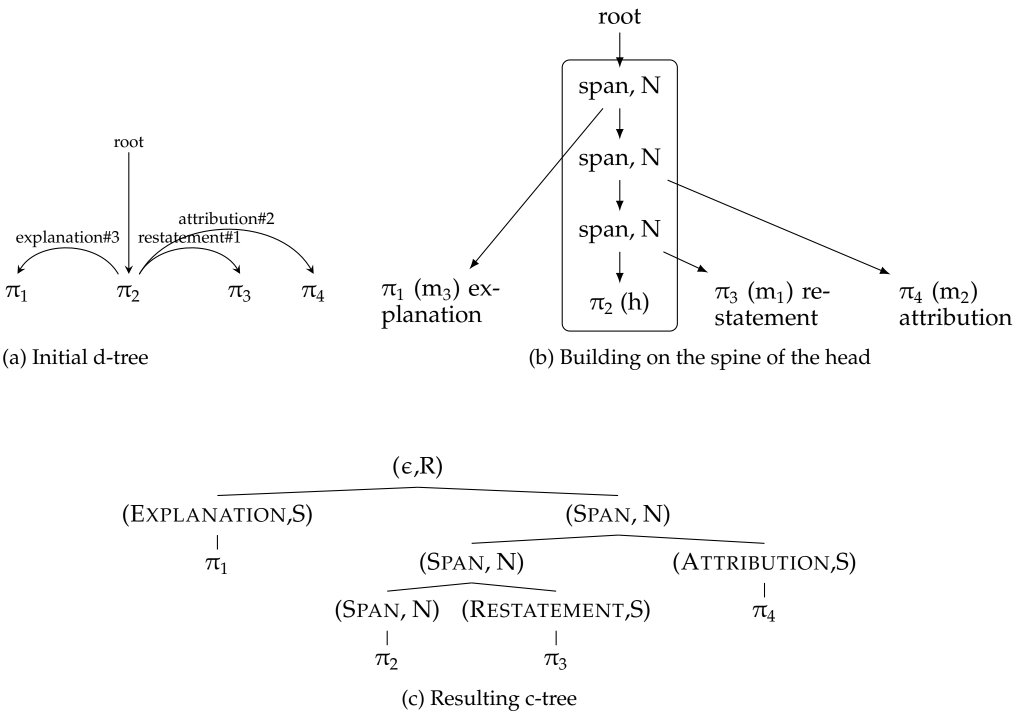
Algorithm 2 Head-ordered dependency tree to headed RST tree T_h translation

```
Input: weakly-ordered RST d-tree $\mathcal{D}$
Output: RST c-tree $\mathcal{C}$
  Nodes := GETPOSTORDERTRAVERSAL($\mathcal{D}$)
  for each $h \in$ Nodes do
      Add c-node $v := \langle\langle span, Nucleus\rangle, h, \{h\}\rangle$                ▷ pre-terminal
      $\phi(h) := v$
      Sort $M_h(\mathcal{D})$, yielding $\bar{M}_h^1 \prec_h \bar{M}_h^2 \prec_h \ldots \prec_h \bar{M}_h^J$
      for each $j = 1, \ldots, J$ do
          Add c-node $v := \langle\langle span, Nucleus\rangle, h, \mathcal{I} \cup \bigcup_{\bar{M}_h^j} \mathcal{J}_m\rangle$ to $\mathcal{C}$
          Obtain c-node $\phi(h) = \langle X, h, \mathcal{I}\rangle$
          for each $m \in \bar{M}_h^j$ do
              Obtain c-node $\phi(m) = \langle Y_m, m, \mathcal{J}_m\rangle$ ▷ subtree headed by $m$ already built because of the post-order traversal
              Let $Z$ be the label of the arc $\langle h, m, Z\rangle$
              $Y_m := Z$
              if $Y_m == \langle r_m, Nucleus\rangle$ then ▷ multinuclear relation: $h$ gets the same label as $m$
                  $X := Y_m$
              end if
          end for
          Set $\phi(h)$ and $\{\phi(m) | m \in \bar{M}_h^j\}$ as children of $v$
          $\phi(h) := v$
      end for                    ▷ the tree headed by h grows from first attachment to last
  end for
  Let $v = \langle X, h, \{1, \ldots, L\}\rangle$ be the root of $\mathcal{C}$, set $X := \langle \epsilon, Root\rangle$
```

We can now establish a correspondence between weakly ordered RST d-trees and RST c-trees.

Proposition 1
Unaryless headed RST c-trees and weakly-ordered RST d-trees are isomorphic.

Proof. We use the construction in Figure 7, adapted from Fernández-González and Martin (2015). Let $\mathcal{D}$ be given. We visit each node $h \in \{1, \ldots, L\}$ and split it into $J + 1$ nodes, where J is the number of equivalence classes of the K modifiers, organized as a linked list. For each class of equivalence on the modifiers $\bar{M}_h^j$, $j \in \{1, \ldots, J\}$, (i) let h be the head EDU of the $(J + 1 - j)$-th node of the linked list, (ii) for each modifier $m_k \in \bar{M}_h^j$, (a) move the tail of the arc $\langle h, m_k, Z_k\rangle$ to the $(J + 1 - j)$-th node of the list, (b) move the head of this arc to the highest node headed by EDU m_k $\phi(m_k)$ and assign the label Z_k to $\phi(m_k)$. Because the incoming and outgoing arcs of the linked list component are the same as in the original node h, the tree structure is preserved. After doing this for every h, add the leaves and propagate the yields bottom up. It is straightforward to show that this procedure yields a valid unaryless, headed RST c-tree. Because there is no loss of information (the orders $\prec_h$ are implied by the order of the nodes in each



(a) Initial d-tree

(b) Building on the spine of the head

(c) Resulting c-tree

Figure 7
Transformation of a strictly ordered RST d-tree (a) into a binary RST c-tree (c) using the order of attachment (b).

spine), this construction can be inverted to recover the original RST d-tree. Conversely, if we start with a unaryless headed RST c-tree, traverse the spine of each h, and attach the modifiers in each equivalence class $\bar{M}_h^1, \ldots, \bar{M}_h^J$ in order, we get a weakly ordered RST d-tree (also an invertible procedure). ■

If we constrain each equivalence class to contain exactly one member, we get a strong order relation on the modifiers of each head EDU. The resulting structure is called a **strongly ordered RST d-tree** and it is straightforward to show it is isomorphic to a binary headed, RST c-tree. This variant is of particular interest given the widespread use of binarized RST trees for the training and evaluation of existing discourse parsers.

Binarization is not required to convert from a constituency to a dependency tree, however. In fact, the RST d-tree $\mathcal{D}$ obtained from converting a headed RST c-tree $\mathcal{C}$ is structurally identical to the RST d-tree $\mathcal{D}_{LB}$ obtained from first applying a left-heavy binarization to $\mathcal{C}$ then converting the binary c-tree. The two d-trees $\mathcal{D}$ and $\mathcal{D}_{LB}$ differ only in their order of attachment: In $\mathcal{D}$, several modifiers of the same head can have the same order, whereas by construction all modifiers of the same head have distinct orders in $\mathcal{D}_{LB}$. Direct conversion (without binarization) produces a d-tree that is strictly equivalent to the original headed RST c-tree. This preserves the exact semantics of the original annotation, not only for constituency nodes that dominate more than two nuclei (n-ary multinuclear relations) but also for constituency nodes that dominate more than one satellite. This configuration corresponds to several independent mononuclear relations, where each satellite independently modifies the nucleus. While independent modifications by several mononuclear relations are extremely rare in the RST-DT corpus, such configurations have been advocated for and used more extensively in other corpora (van der Vliet and Redeker 2011).

When a right-heavy binarization is applied, n-ary nodes for multinuclear relations are replaced with $n - 1$ binary nodes where each nucleus is the head of the recursive binary node on its right. In the corresponding dependency tree, this translates to a chain of $n - 1$ dependencies between the linearly ordered nuclei of the relation. When no binarization or the left-heavy binarization procedure is applied, the first nucleus is the head of all other nuclei. In the corresponding dependency tree, the first nucleus has the $n - 1$ other nuclei as its dependents. We will refer to the composition of the right-heavy binarization and conversion to dependency as the **chain** transformation, and the direct conversion of n-ary c-trees to dependency as the **tree** transformation. We leave aside the left-heavy conversion procedure that does not appear to be used in the RST parsing literature.

In theory, predicting a head-ordered dependency tree resembles predicting a dependency tree together with the rank of attachment of each dependent to its governor. In practice, Fernández-González and Martins (2015) reduced these two problems into a standard dependency parsing problem by concatenating relation label and attachment rank into a unique label. Their experiments showed that the increase of the number of classes for prediction did not have a negative impact on the performance of their syntactic parser.

3.5 Dependency-Based Evaluation

Having defined precisely the correspondences between constituent and dependency structures, we can now discuss how dependency structures should be evaluated.

In syntactic parsing evaluation, dependency structures are compared straightforwardly with respect to their edges, either labeled or unlabeled. Unlabeled attachment score (UAS) is the proportion of correct (oriented) edges in a system prediction with respect to edges in the gold dependency structure. Labeled attachment score (LAS) is the proportion of correct oriented edges with the correct relation label in the gold. These evaluations are also types of edit distances and provide true metrics over dependency trees. These two measures can be directly applied to discourse representations in the form of a labeled dependency structure between EDUs. When the structure is constrained to be a tree spanning n given discourse units, the number of dependencies is necessarily $n - 1$ both in the gold and the predicted tree structure. This is not necessarily the case in discourse theories which allow for non-tree graphs, such as SDRT in the corpus used in Muller et al. (2012), where precision and recall are used.

In the case of headed RST c-trees, we have seen that there are several options to produce a comparable dependency structure, though the notion of headedness is not part of the original conception of an RST constituent tree. We have shown the equivalence with head-ordered dependency trees, and it is thus necessary to account for the rank of attachment of dependents to their head.

We do not know of any work considering this kind of structure in the discourse dependency parsing literature, as predicted dependencies approximate more complex structures: in the case of RST parsing (Li et al. 2014), human annotations are converted to simple dependency structures, and then compared to predicted structures with UAS and LAS, without considering attachment ranks. They also give constituent scores based on a list of correspondences between simple dependency trees and simple constituent trees, but they do not explain their conversion further, neither how they resolve the inherent ambiguity of simple dependency trees.

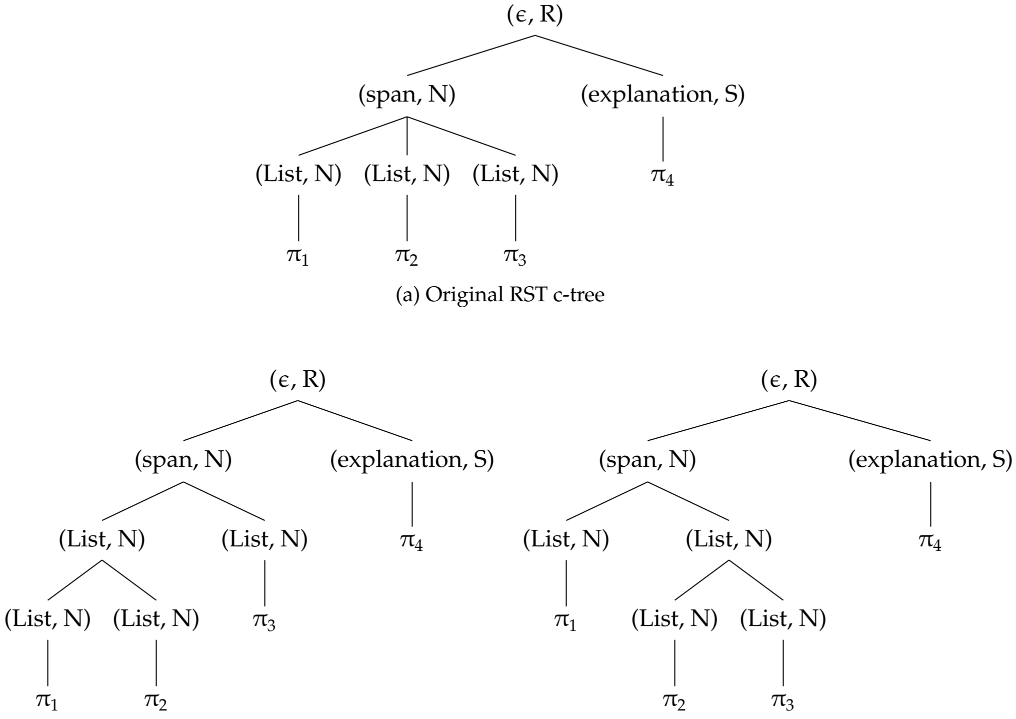
Some researchers argue that dependency metrics, even on the simple version without attachment order, are a better reflection of the decisions made during the annotation process than constituency metrics, where a unique attachment decision counts several times. Some attachment rankings are not semantically meaningful but are rather an artifact of the annotation process (Iruskieta, da Cunha, and Taboada 2015, page 276), especially in the case of binary trees (van der Vliet and Redeker 2011, page. 8). Indeed, the "qualitative metric" proposed by Iruskieta, da Cunha, and Taboada (2015) seems very similar to the standard dependency metrics (UAS/LAS). If we consider again Figure 5, for instance, we can see that the decision to include π_3 in the scope of the explanation between π_1 and π_2 in (c) is really an attachment decision about the pair (π_2, π_3), which makes the whole subtree false under constituency metrics with respect to the tree in (b), while dependency metrics will evaluate rightly the explanation between π_1 and π_2. The binarization of trees compounds the problem by forcing artificial constructions for relations with arity of more than 2, which can be seen later in Figure 8 for the List relation, where the two options (left or right binarization) are semantically equivalent and a matter of convention, but can generate errors in the constituent evaluations, while a simple dependency representation would make them truly equivalent. This cannot be solved by post-processing these subtrees without losing potential List substructures (something that would be lost on simple dependency trees, too). Note that sublists could be represented with a head-ordered dependency tree, thus keeping the binary relation framework that is assumed by all parsing approaches. All of this militates in favor of using dependency metrics, as a better reflection of the annotation decisions, and a more cautious approach to the semantics of discourse relations.

3.6 Dependency-Based and Constituent-Based Metrics Compared

How do the original measures on RST c-structures compare with evaluations of d-structures? Our answer is that the original measures are not helpful in gauging the accuracy of the semantic content of a candidate c-structure vis a vis a reference structure, as we will show below. Once we add heads to c-structures, however, we can prove equivalences between measures for headed c-trees and d-structures, exploiting the result in Section 3.3 that dependency structures in which we keep track of the order of attachment are isomorphic to headed RST constituent trees. We can then use UAS/LAS measures on RST constituent trees with only minor modifications. We will indicate how a combination of the UAS measure with a measure on the order of attachment to the head (OUAS) can capture spans and headedness in headed RST trees; we can extend such a measure to then evaluate a candidate tree with respect to a reference tree concerning the relation and nuclearity decorations. Thus, the constituency-based measures on headed trees can all be reproduced within the dependency framework.

With regard to the original measures proposed for non-headed c-trees, we note that span by itself is not a very informative measure semantically. Two trees may define the same spans with subtrees yet have completely discordant d-structures: consider the ordered d-trees

(2) $\{a \leftarrow_3 c, b \leftarrow_1 c, c \rightarrow_2 d\}$ and $\{a \leftarrow_3 b \rightarrow_1 c, b \rightarrow_2 d\}$



(a) Original RST c-tree

(b) Left-heavy binarization

(c) Right-heavy binarization

Figure 8
Left- vs. right-heavy binarization strategies for RST c-trees.

The two d-trees in Example (2) do not agree on any unlabeled edges yet they yield c-structures with the same spans. Given that we are interested semantically in what EDUs have discourse or rhetorical connections to other discourse units, the Span measure by itself risks, as this example makes evident, to tell us nothing, or next to nothing. Spans of subtrees in an RST c-tree do not even determine an undirected, unlabeled d-tree, because nuclearity information, together with an interpretive principle that links nuclearity to headedness, is needed to determine the d-tree's head. Consider the following RST c-tree: $[a, [b, c]]$ without any nuclearity decoration. One d-tree that could correspond to this is $a \rightarrow b \rightarrow c$ if the nuclearity decoration is $[a_N, [b_N, c_S]_S)$, but another is $\{c \rightarrow a, c \rightarrow b\}$ if the nuclearity decoration is $[a_S, [b_S, c_N]_N]$. So what does the span measure tell us about? It does tell us about higher order RST constituents; but as we have been at some pains to emphasize, these higher-order RST constituents are not semantically transparent, and so a measure that tells us mainly about them is not what we should be interested in as discourse theorists. This point is further reinforced by noting the same d-tree, with its semantically important discourse relations between constituents, can give rise to two different c-trees with different spans in spider configurations. The spans then are semantically epiphenomenal, at best. A similar point can be made about nuclearity. Finally, if we add information about relation to spans—for instance $[a_N, b_S, Explanation]$, we know for instance that a relation like Explanation holds of subspans of a and b but once again we are far from determining a full discourse structure.

Once we add the notion of headedness to c-trees, however, there are strong correlations between measures on headed c-trees and measures on d-structures, as expected.

By combining a measure of span and head accuracy, we can capture unlabeled accuracy in some cases. Consider equating a measure of unlabeled accuracy with a measure, δ_{SH}, over triples (a, b, c), where a is the leftmost EDU in a span, b its rightmost EDU, and c the EDU that is the head of that span (or tree with that span). Where T, T' are two headed trees, $\delta_{SH}(T, T') =$ the number of triples in T that do not occur in T'. δ_{SH} is an evaluation measure that strongly resembles δ_{uas}, which also counts the number of directed edges in the target tree missing in the candidate tree.

Now, given the isomorphism $\mathcal{I}$: *headed-c-tree* $\longrightarrow$ *Ordered-d-tree* defined in the proof of Proposition 1, we can show that there is a strict correspondence between the heads and spans of subtrees of a headed c-tree and the directions of the arrows in a d-tree: If the head of a tree T_0 is t_0, then the arrows will flow from t_0 to other elements of T_0, and vice-versa. Hence any mistake in a head on a span defined by T will translate to a mistake on the direction of an arrow in $\mathcal{I}(T)$ and hence an increase in $\delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$. To go back to our case in Example (2), we see that the candidate tree has three errors on what the head of a span is, which correspond to the three UAS arrow errors. On the other hand, if the heads of spans are all correct, then the spans will determine correct attachments.

However, to furnish an equivalent dependency based measure to our δ_{SH}, we need also to consider the order in which dependents are attached to the head. If two d-trees have the same head, we can compare the strings of order operations used in each one to generate the tree. We will use an edit distance on these strings, in which we count the wrong choices among the possibilities for attaching dependents to their heads made in constructing the candidate tree with respect to the reference tree. This distance can thus be added to a UAS count to produce a OUAS evaluation measure, δ_{ouas}, that is provably equivalent to δ_{SH}. For example given a reference ordered d-tree $\{a \leftarrow_2 b, b \rightarrow_1 c, a \rightarrow_3 d\}$, where b is the head, the candidate tree $\{a \leftarrow_1 b, b \rightarrow_2 c, a \rightarrow_3 d\}$ has exactly the same arrows and so hence a perfect UAS score. However, the bracketed representation of the two resulting c-trees $[[a[b, c]], d]$ and $[[a, b], c], d]$, shows that in building the candidate tree we made one wrong choice (attaching b first to a), which leads under the translation to a wrong subtree and span (a b instead of b c). If we look at the strings of applications to b, we have $f_{\rightarrow d}(f_{\leftarrow a}(f_{\rightarrow c}(b)))$ for the reference tree and $f_{\rightarrow d}(f_{\rightarrow c}(f_{\leftarrow a}(b)))$ for the candidate; in counting the wrong choices made, we see that we made one wrong choice of applying $f_{\leftarrow a}$ instead of $f_{\rightarrow c}$; counting such wrong choices would assign the same distance to the candidate tree as would δ_{SH} in this case. That is, δ_{SH} and δ_{ouas} agree. In Appendix A we sketch a proof of the following proposition:

Proposition 2
Let T and T' be two, projective headed c-trees over a common set of EDUs $\{b_1, \ldots, b_n\}$. Then $\delta_{SH}(T, T') = \delta_{ouas}(\mathcal{I}(T), \mathcal{I}(T'))$.

With this proposition it is easy to see how to extend δ_{ouas} to capture constituent measures that also evaluate predictions of nuclearity or relation.

Although we have shown how evaluation measures on dependency trees can express various measures on headed trees, we believe that there are conceptual reasons to prefer simple dependency measures like δ_{uas} and δ_{las} over a measure like δ_{ouas}; while differences in order accurately reflect differences in higher order constituents of two c-trees, the semantic relevance of these higher order constituents remains unclear for reasons that we gave in the last section. Hence, we do not see a compelling reason to try to capture them, and hence we do not need to move to a measure like δ_{ouas} from δ_{uas}.

4. Evaluation

In this section, we turn to the empirical study of RST parsing, using both the dependency and the constituent viewpoint in a unified framework. This is made possible by the explicit conversion between viewpoints, and the awareness of the specificities of theoretical models and their practical representations.

4.1 A Diverse Panel of Parsing Strategies

We evaluate the performance of a diverse sample of RST discourse parsers on the test set of the RST-DT, using manual EDU segmentation. We build on the work of Morey, Muller, and Asher (2017), who collected and harmonized the c-trees predicted by nine RST (constituency) parsers and implemented several variants of the RST-Parseval (constituency) metrics. We add to this collection the d-trees output by the MST parser of Hayashi, Hirao, and Nagata (2016), which we augment with the nuclearity and order of attachment to the head (predicted by heuristic components described below). We do not present results for the dependency parser of Li et al. (2014) and the constituency parser of Li, Li, and Hovy (2014), because we were unable to obtain or reproduce usable predictions from them.

Because there is such an imbalance between parsers representative of constituent and dependency approaches, we add an in-house dependency parser to the comparison to increase the diversity of parsing strategies. We built this parser from standard elements: (i) an Eisner decoder (Eisner 1996) that outputs projective dependency trees, classic in syntactic dependency parsing (McDonald, Crammer, and Pereira 2005) and already used in Li et al. (2014); (ii) a subset of the lexical, syntactic, and organizational features commonly used in the aforementioned constituency-based RST discourse parsers; (iii) logistic regression models with L_2 regularization (also known as Maximum Entropy) (Nigam, Lafferty, and McCallum 1999): binary classifiers for attachment and multi-class classifiers for labeling. The models use the same features as most feature-heavy constituent approaches up to the syntactic level. We follow Reitter (2003), Joty, Carenini, and Ng (2015), and Feng and Hirst (2014) and decompose the parsing process in two stages with distinct models for intra- and inter-sentential relations. Our parser builds an RST dependency tree for each sentence, then builds a document-level dependency tree by adding dependencies between the heads of the sentence-level trees. The nuclearity of each relation occurrence is determined heuristically in a post-processing stage: It is multinuclear (the dependent is labeled “Nucleus”) if all occurrences of this relation type are multinuclear in the training section of the RST-DT, else it is mono-nuclear (“Satellite”). We train two variants of this dependency parser, respectively, on the *chain* and *tree* transformations of the RST-DT described in Section 3. The two variants are trained on different dependency trees but they use the exact same components, except for a slight adaptation of the heuristics used to determine the order of attachment to the head (described below).

A key difference compared with previous studies that included dependency parsers for discourse (Muller et al. 2012; Li et al. 2014; Hayashi, Hirao, and Nagata 2016) is that we consider *head-ordered* dependency trees, as presented in Section 3.3. This enables us to produce equivalent RST c-trees for comparison with existing constituency-based RST parsers. Although the order of attachment could be learned along with the relation label as in the syntactic work of Fernández-González and Martins (2015), this would multiply the number of categories, on a data set that is much smaller than syntactic parsing data sets. So instead we determine the order of attachment heuristically, in a post-processing phase. The modifiers on each side (left, right) of an EDU are ordered from the inside out. If a head h has a SAME-UNIT modifier m_{su} on its right, all modifiers from h to m_{su} included are attached before other modifiers are considered. The relative order of the remaining left and right modifiers is determined by their distance to the head, in number of sentences then EDUs (lower is attached first), and right modifiers are attached before left ones. Finally, for the parser trained on the "tree" transformation only, modifiers of the same head with the same multinuclear relation label are assigned the same order to form n-ary multinuclear relations.

Our discourse parser for RST thus blends features and decoding strategies from existing constituent and dependency parsers. We decompose the prediction problem for RST discourse parsing into three problems: labeled dependency parsing, prediction of nuclearity, and prediction of the order of attachment to the head, where the latter two stages depend on the first one. Even if nuclearity and order could be modeled as independent learning problems or jointly with relation labeling, our heuristics for them provide a solid, inexpensive baseline that can additionally be applied directly to the output of other dependency parsers for RST (Li et al. 2014; Hayashi, Hirao, and Nagata 2016). Keeping nuclearity and order separate from labeled dependency parsing further enables our parser to generalize well to other formalisms like SDRT.

4.2 Grouping Parsing Strategies

In line with the standard dependency metrics for syntactic parsing, we defined and implemented several metrics on the dependencies of head-ordered RST d-trees: unlabeled attachment score (UAS), labeled attachment score where the label is restricted to nuclearity (LAS-N), relation (LAS-R), or includes both (LAS-F). The source code of the new parsers and evaluation procedures is available online.[4]

We distinguish three groups among RST parsers. The first group consists of constituency parsers that do not use the notion of head EDU. **HHN16 HILDA** is a reimplementation by Hayashi, Hirao, and Nagata (2016) of the HILDA parser (duVerle and Prendinger 2009; Hernault et al. 2010), which was one of the first text-level discourse parsers to be published that was trained and evaluated on the RST-DT. It is a greedy bottom–up parser that uses two support vector machine models to predict structures and their labeling, on a set of lexical, syntactic, and structural features. **SHV15 D** follows the same architecture as HILDA, with minor differences in the models (perceptron for structure, logistic regression for labeling) and features (syntactic features expressed on dependency trees) (Surdeanu, Hicks, and Valenzuela-Escárcega 2015). **JCN15 1S-1S** is a two-stage CKY parser that first builds sentence-level subtrees then combines them into a document-level RST tree (Joty, Carenini, and Ng 2015). To each stage corresponds a distinct Dynamic conditional random field (CRF) model that jointly estimates the structure and labeling of constituents. **FH14 gCRF** is a two-stage greedy parser where each stage leverages two linear-chain conditional random field (CRF) models to predict structure and labeling (Feng and Hirst 2014). **LLC16** is a CKY parser that uses a hierarchical bidirectional LSTM model with attention mechanism to learn representations of text spans, complemented with a minimal set of lexical, syntactic, and structural features (Li, Li, and Chang 2016).

[4] https://github.com/irit-melodi/irit-rst-dt.

The second group consists of constituency parsers that make crucial use of the notion of head EDU, introduced in the shift-reduce RST parser of Sagae (2009).[5] In particular, the representations of (complex) discourse units are built from the representations of their head EDUs. **JE14 DPLP** is a shift-reduce parser that jointly learns to predict transitions and to project the bag-of-word representation of each EDU into a latent space, complemented with a minimal set of lexical, syntactic, and structural features introduced in Ji and Eisenstein (2014) and subsequently improved in an unpublished version by the authors. **BPS16** is a sequence-to-sequence parser that uses a hierarchical bidirectional LSTM model to learn representations of text spans in a multi-task learning setting (Braud, Plank, and Søgaard 2016). One of the auxiliary tasks consists in predicting binary dependencies extracted from the RST c-trees by means of a procedure akin to that of Li et al. (2014) evoked in Section 3.1. **BCS17 mono** and **BCS17 cross** are two variants (mono- and cross-lingual, respectively) of a shift-reduce parser that uses a feed-forward neural network to learn representations for EDUs from structural, syntactic, and lexical features (Braud, Coavoux, and Søgaard 2017).

The third group consists of dependency parsers for RST whose output can be converted to RST c-trees. **HHN16 MST** is a parser that uses a Maximum Spanning Tree decoder and a linear model trained with MIRA, on lexical, syntactic, and structural features (Hayashi, Hirao, and Nagata 2016). It is trained on dependency trees obtained by applying the conversion procedure of Li et al. (2014) on right-heavy binarized c-trees from the RST-DT. **dep-chain** and **dep-tree** are two variants of our proof-of-concept parser that uses an Eisner decoder and logistic regression models for intra- and inter-sentential attachment and labeling, as described above. The first variant is trained on d-trees produced from right-heavy binarized c-trees and the second is trained on d-trees produced from the original, n-ary c-trees from the RST-DT. The dependency trees output by these three parsers (HHN16 MST, dep-chain, and dep-tree) originally contain neither the nuclearity status of relations nor the order of attachment. We apply the post-processing heuristics described above to add both types of information and obtain fully-specified head-ordered RST d-trees.

4.3 Evaluations of Parsers Against Right-Heavy Binarized c-trees from the RST-DT

We first replicated the de facto standard evaluation protocol from the literature and use as reference a right-heavy binarized version of the c-trees from the RST-DT. In Table 1, we report the F_1 scores of the standard constituency metrics in RST-Parseval: unlabeled spans (S), spans labeled with Nuclearity (N), Relation (R), or both (F), following the abbreviations in Hernault et al. (2010). Additionally, we report the scores for the same metrics where each span is augmented with its head EDU, denoted S+H, N+H, R+H, and F+H.

The overall level of performance on RST-Parseval is quite homogeneous among the three groups of parsers, even though two parsers stand out: FH14 gCRF on unlabeled spans (S) and spans labeled with nuclearity (N), JE14 DPLP on spans labeled with relation (R), and fully labeled spans (F). Another point is that the dependency parsers (HHN16 MST, dep-chain, dep-tree) are competitive on constituency metrics with constituency parsers. This is slightly surprising because these dependency parsers are trained toward a different objective (attachments between head EDUs), and we

[5] In each case, the head is determined via the nuclearity principle, i.e., following the nucleus, or is the leftmost EDU in case of a non-mononuclear relation.

Table 1
RST-Parseval F_1 scores against right-heavy binarized reference trees. S = Span; N = (Span and Nuclearity); R = (Span and Relation); F = (Span and Relation and Nuclearity). An asterisk (*) signals scores on predictions we reproduced using code and material made available by the authors; a double asterisk (**) signals scores on predictions provided by the author with an improved, unpublished version of the parser posterior to the original study; *+h* marks the use of our heuristic component to determine the nuclearity and order of attachment. The best score in each group is underlined; the best score overall is in **bold**.

parser	S	N	R	F	S+H	N+H	R+H	F+H
LLC16	82.2	66.5	51.4	50.6	78.9	64.5	49.9	49.2
HHN16 HILDA	82.6	66.6	54.6	54.3	79.3	64.9	53.3	53.0
SHV15 D *	82.6	67.1	55.4	54.9	79.3	64.9	53.7	53.2
JCN15 1S-1S	82.6	68.3	55.8	55.4	79.8	66.4	54.6	54.2
FH14 gCRF *	**84.3**	**69.4**	56.9	56.2	**80.4**	**66.7**	55.1	54.5
BPS16	79.7	63.6	47.7	47.5	76.0	61.1	46.4	46.1
BCS17 mono	81.0	67.7	55.7	55.3	78.8	66.3	54.6	54.2
BCS17 cross	81.3	68.1	56.3	56.0	78.9	66.4	55.0	54.7
JE14 DPLP **	82.0	68.2	**57.8**	**57.6**	79.5	66.6	**56.6**	**56.5**
HHN16 MST +h	82.8	67.6	54.5	53.8	79.6	65.8	53.6	52.9
dep-tree +h	81.4	68.3	55.2	54.7	79.0	66.6	54.2	53.7
dep-chain +h	81.3	68.3	55.2	54.7	78.8	66.6	54.1	53.6

only used a heuristic component to predict the order of attachment between modifiers of the same head. The four columns on the right in Table 1 show the scores obtained with the same evaluation procedure, but where each span is augmented with the index of its head EDU. Although the relative performance of parsers is globally stable inside each group of parsers, the addition of the head EDU takes a slightly higher toll on the first group of parsers compared with the second and third groups.

The scores in Table 1 suffer from a bias we described in Section 2.4: Marcu's encoding of RST into c-trees and variant of Parseval give an arguably higher score than is warranted. To obtain more accurate scores, we use the alternative encoding of RST c-trees represented in the top row of Figure 3 and apply a variant of RST-Parseval that includes the root node but excludes pre-terminals (and leaves)—namely, the EDUs. Each node in these trees represents a labeled attachment decision made by an RST parser, and the evaluation procedure needs to consider only half the number of nodes compared to Marcu's RST-Parseval. The scores for this alternative RST-Parseval are given in Table 2. The scores on unlabeled spans (S) are 15 to 20 points lower than in Table 1. In fact, the alternative scores on unlabeled spans (the S column in Table 2) directly correspond to the non-trivial fraction of the original scores (the S column in Table 1), where manually segmented EDUs make up for half the number of unlabeled spans. A visible consequence is that differences between parsers on the S metric are doubled, for instance FH14 gCRF is now 3 points ahead of the second best parser (HHN16 MST). This relationship is less obvious for labeled spans (N, R, F) because the alternative encoding irons out several factors at once: trivial spans, but also redundant information in the node labels about nuclearity and relations (multinuclear relations, SPAN labels for mono-nuclear relations), as detailed in Section 2.4. The alternative scores are 10 to 20 points lower than the original scores, but all parsers are not equally

Table 2
Corrected constituency evaluation on labeled attachment decisions.

parser	S	N	R	F	S+H	N+H	R+H	F+H
LLC16	64.5	54.0	38.1	36.6	57.4	52.2	36.1	35.7
HHN16 HILDA	65.1	54.6	44.7	44.1	58.4	52.6	43.0	42.8
SHV15 D *	65.3	54.2	45.1	44.2	58.2	52.2	43.1	42.9
JCN15 1S-1S	65.1	55.5	45.1	44.3	59.3	**54.2**	43.5	43.3
FH14 gCRF *	**68.6**	**55.9**	45.8	44.6	**60.5**	53.8	43.4	43.1
BPS16	59.5	47.2	34.7	34.3	51.7	45.7	33.6	33.5
BCS17 mono	61.9	53.4	44.5	44.0	57.4	52.2	43.4	43.2
BCS17 cross	62.7	54.5	45.5	45.1	57.5	52.8	43.8	43.8
JE14 DPLP **	64.1	54.2	**46.8**	**46.3**	58.7	53.3	**45.7**	**45.6**
HHN16 MST +h	65.6	53.2	43.5	42.5	58.0	51.4	41.9	41.7
dep-tree +h	62.7	53.2	43.4	42.4	57.5	51.8	42.1	41.6
dep-chain +h	62.6	52.9	43.4	42.4	57.4	51.6	42.2	41.7

affected, which results in a slightly different picture. On spans labeled with nuclearity (N), the best system FH14 gCRF is only 0.4 points ahead of JCN15 1S-1S, down from 1.1 points; HHN16 HILDA now appears at the same level as BCS17 cross (ahead by 0.1 points), while its original score was 1.5 points lower. The best parser on R and F, JE14 DPLP, maintains its lead but the second best parser on F is now BCS17 cross. The change of encoding and evaluation procedure widens the gap in performance between constituency and dependency parsers on labeled spans. Note, however, that this gap could be reduced, on all metrics, by substituting the heuristic component we used to predict the order of attachment with a machine learning model. The greedy and chart-based constituency parsers appear to be better at predicting spans than the shift-reduce and dependency parsers. This reflects the more explicit focus in their modeling and training objective on predicting accurate boundaries between (complex) discourse units, while dependency parsers and shift-reduce constituency parsers focus more on the relationship between head EDUs.

Dependency metrics offer a complementary point of view on parsers' performance. Table 3 provides dependency scores for a subset of possible metrics: on unlabeled dependencies (UAS), dependencies labeled with nuclearity alone (LAS-N), relation (LAS-R), or both (LAS-F).

On unlabeled dependencies, the best performing systems are shift-reduce constituency parsers (JE14 DPLP, BCS17 mono and cross) and dependency parsers (dep-chain, dep-tree). This is hardly surprising given the fact that shift-reduce constituency parsers are trained to predict both a constituency and a dependency structure. JE14 DPLP is the best system on all dependency metrics, with a slight advantage on unlabeled dependencies (+0.2 points compared to dep-chain on UAS) and a marked advantage on labeled dependencies, especially when the relation is included (0.9 points ahead of dep-chain on LAS-N, 2.1 and 2.2 points ahead of BCS17 mono on LAS-R and LAS-F). This provides empirical support to the claims made by Ji and Eisenstein (2014) that their vector-space representation of EDUs improves on existing approaches for the identification of discourse relations and, to a lesser extent, nuclearity status.

Table 3
Dependency evaluation. UAS = unlabeled dependencies; LAS-N = dependencies labeled with nuclearity alone; LAS-R = dependencies labeled with relation alone; LAS-F = dependencies labeled with relation and nuclearity.

parser	UAS	LAS-N	LAS-R	LAS-F
LLC16	62.9	58.5	42.1	41.5
HHN16 HILDA	64.7	59.1	49.1	48.8
SHV15 D *	64.2	59.0	49.5	49.2
JCN15 1S-1S	65.8	60.5	48.9	48.6
FH14 gCRF *	67.6	61.7	50.1	49.8
BPS16	61.7	55.0	40.6	40.5
BCS17 mono	68.2	62.1	51.6	51.4
BCS17 cross	67.9	62.4	51.0	50.9
JE14 DPLP **	**69.2**	**63.6**	**53.7**	**53.6**
HHN16 MST +h	66.6	59.9	48.8	48.4
dep-tree +h	68.0	62.1	49.8	49.2
dep-chain +h	69.0	62.7	50.8	50.2

4.4 Evaluation Against the Original Non-binarized RST Trees

We also looked at constituency and dependency metrics against the original, non-binarized c-trees from the RST-DT. Table 4 provides the scores using Marcu's encoding and evaluation procedure, Table 5 the scores on dependency metrics. The relative performance of parsers is quite similar to that observed against right-heavy binarized c-trees. The scores of greedy and chart-based parsers are stable compared to Tables 1–3, while the shift-reduce and dependency parsers obtain lower scores. The performance of parsers is more homogeneous overall. The d-trees produced by the *tree* transformation appear to be harder for parsers than those produced by the *chain* transformation. Note

Table 4
RST-Parseval metrics scores against the original (non-binarized) trees.

parser	S	N	R	F	S+H	N+H	R+H	F+H
LLC16	82.6	66.9	51.8	51.1	79.2	64.9	50.4	49.7
HHN16 HILDA	82.7	66.9	54.8	54.5	79.4	65.2	53.5	53.2
SHV15 D *	82.5	67.0	55.3	54.8	79.2	64.8	53.7	53.2
JCN15 1S-1S	82.5	68.4	55.9	55.5	79.8	66.5	54.7	54.3
FH14 gCRF *	**84.4**	**69.6**	57.1	56.4	**80.5**	**66.9**	55.3	54.7
BPS16	79.6	63.4	47.5	47.2	75.8	60.9	46.1	45.9
BCS17 mono	80.6	67.6	55.5	55.2	78.5	66.1	54.5	54.1
BCS17 cross	81.0	67.9	56.2	55.9	78.6	66.2	54.8	54.5
JE14 DPLP **	81.5	67.7	**57.3**	**57.2**	78.9	66.1	**56.2**	**56.0**
HHN16 MST +h	82.5	67.4	54.3	53.6	79.3	65.6	53.3	52.6
dep-tree +h	81.3	68.4	55.4	54.9	78.8	66.8	54.3	53.8
dep-chain +h	80.8	68.0	55.0	54.5	78.3	66.3	53.9	53.4

Table 5
Dependency evaluation on non-binarized trees.

parser	UAS	LAS-N	LAS-R	LAS-F
LLC16	62.7	58.4	42.1	41.5
HHN16 HILDA	63.1	57.8	47.9	47.7
SHV15 D *	62.4	57.7	48.4	48.1
JCN15 1S-1S	64.4	59.3	48.0	47.7
FH14 gCRF *	65.6	60.0	48.5	48.2
BPS16	59.7	53.1	38.7	38.6
BCS17 mono	66.1	60.3	49.9	49.7
BCS17 cross	65.6	60.4	49.1	48.9
JE14 DPLP **	**66.9**	**61.7**	**51.7**	**51.7**
HHN16 MST +h	64.2	58.0	47.1	46.6
dep-tree +h	66.3	60.9	48.7	48.0
dep-chain +h	66.4	60.6	48.7	48.1

that our *chain* and *tree* parser obtain similar scores, despite the fact that the former is trained on a different transformation of the reference trees. This suggests that the higher level of accuracy attainable by the *chain* parser that trains on shorter dependencies compensates for the mismatch between the distributions of attachments during the training and testing stages.

4.5 Pairwise Similarity Between Parser Predictions

The combined use of alternative constituency and dependency metrics has provided us with a richer, more nuanced perspective on the relative performance of RST discourse parsers and the diversity of their predictions. In an attempt to get better insight into their structural similarity, we computed pairwise similarity between the predictions of parsers. We compute constituency and dependency metrics with the same evaluation procedures as before, taking the prediction of each parser in turn as the reference. In line with the standard setting used for training and evaluating parsers in the literature, we again consider right-heavy binarized RST c-trees.

The pairwise similarity scores in Table 6 tend to confirm the existence of two clusters of parsers suggested earlier: parsers from the first group in one cluster, and parsers from the second and third group in another. Pairwise similarity scores tend to be higher inside a cluster than between clusters. For each parser, similarity scores also tend to be higher inside its cluster than with the reference trees. Parsers using the same architecture and core features are the most similar: dep-tree and dep-chain (91.7 on S), followed by the two parsers BCS17 mono and cross (81.5 on S). The dep-chain parser exhibits a relatively high level of similarity with the shift-reduce constituency parsers (75.9 with BCS17 cross, 75.3 with BCS17 mono, 70.8 with JE14 DPLP). The structural similarity of their predictions empirically confirms the similarity in modeling and architecture between shift-reduce constituency parsers and dependency parsers. Pairwise similarity scores on unlabeled dependencies, reported in Table 7, confirm the clustering of parsers in two groups. The similarity scores between the shift-reduce constituency parsers and the dependency parsers are even higher on dependencies than on constituents. This

Table 6
Pairwise similarity on parser predictions: RST-Parseval S on labeled attachment decisions.

	LLC16	HHN16 H	SHV15 D	JCN15 1	FH14 gC	BPS16	BCS17 m	BCS17 c	JE14	HHN16 M	dep-tr	dep-ch	gold
LLC16	100.0	60.2	62.5	59.8	61.4	57.6	59.4	58.5	58.8	57.3	59.1	58.7	64.5
HHN16 HILDA	60.2	100.0	66.7	62.2	67.8	58.5	60.1	60.4	62.0	61.9	62.9	61.8	65.1
SHV15 D	62.5	66.7	100.0	62.0	65.6	60.5	61.8	62.0	62.7	61.0	61.7	61.9	65.3
JCN15 1S1S	59.8	62.2	62.0	100.0	65.9	58.2	59.5	59.4	60.3	60.8	60.7	60.6	65.1
FH14 gCRF	61.4	67.8	65.6	65.9	100.0	61.4	63.6	63.8	63.3	65.5	64.6	64.7	68.6
BPS16	57.6	58.5	60.5	58.2	61.4	100.0	63.7	62.5	61.0	59.7	62.0	63.6	59.5
BCS17 mono	59.4	60.1	61.8	59.5	63.6	63.7	100.0	81.5	68.8	63.2	71.3	75.3	61.9
BCS17 cross	58.5	60.4	62.0	59.4	63.8	62.5	81.5	100.0	69.8	63.2	72.4	75.9	62.7
JE14	58.8	62.0	62.7	60.3	63.3	61.0	68.8	69.8	100.0	61.5	67.9	70.8	64.1
HHN16 MST	57.3	61.9	61.0	60.8	65.5	59.7	63.2	63.2	61.5	100.0	64.9	65.6	65.6
dep-tree	59.1	62.9	61.7	60.7	64.6	62.0	71.3	72.4	67.9	64.9	100.0	91.7	62.7
dep-chain	58.7	61.8	61.9	60.6	64.7	63.6	75.3	75.9	70.8	65.6	91.7	100.0	62.6
gold	64.5	65.1	65.3	65.1	68.6	59.5	61.9	62.7	64.1	65.6	62.7	62.6	100.0

Table 7
Pairwise similarity on parser predictions: dependency metric UAS.

	LLC16	HHN16 H	SHV15 D	JCN15 1	FH14 gC	BPS16	BCS17 m	BCS17 c	JE14	HHN16 M	dep-tr	dep-ch	gold
LLC16	100.0	59.3	58.5	59.0	59.0	55.5	62.5	61.5	61.6	57.8	61.7	61.8	62.9
HHN16 HILDA	59.3	100.0	67.4	65.8	70.5	61.8	67.9	67.2	68.0	66.0	68.1	69.0	64.7
SHV15 D	58.5	67.4	100.0	65.5	68.2	62.1	67.5	67.5	66.9	63.9	67.2	68.5	64.2
JCN15 1S1S	59.0	65.8	65.5	100.0	70.6	61.7	68.7	67.6	68.5	65.0	69.3	70.1	65.8
FH14 gCRF	59.0	70.5	68.2	70.6	100.0	64.4	71.7	72.3	70.8	68.2	71.6	72.8	67.6
BPS16	55.5	61.8	62.1	61.7	64.4	100.0	66.5	66.1	66.0	61.3	65.1	66.9	61.7
BCS17 mono	62.5	67.9	67.5	68.7	71.7	66.5	100.0	79.5	73.5	68.2	73.5	75.2	68.2
BCS17 cross	61.5	67.2	67.5	67.6	72.3	66.1	79.5	100.0	73.0	67.0	74.2	75.6	67.9
JE14	61.6	68.0	66.9	68.5	70.8	66.0	73.5	73.0	100.0	66.5	72.7	74.3	69.2
HHN16 MST	57.8	66.0	63.9	65.0	68.2	61.3	68.2	67.0	66.5	100.0	70.0	70.9	66.6
dep-tree	61.7	68.1	67.2	69.3	71.6	65.1	73.5	74.2	72.7	70.0	100.0	95.1	68.0
dep-chain	61.8	69.0	68.5	70.1	72.8	66.9	75.2	75.6	74.3	70.9	95.1	100.0	69.0
gold	62.9	64.7	64.2	65.8	67.6	61.7	68.2	67.9	69.2	66.6	68.0	69.0	100.0

reinforces the view that these two families of parsers are closer than usually presented in the RST parsing literature.

5. Conclusion

Text-level discourse parsing has seen a recent surge in experimental approaches using the RST Discourse Treebank for English as an evaluation benchmark. We have seen that representing discourse as constituent structures raises a number of issues, both conceptual and practical. It is compounded by representational choices made in the annotation of the reference corpus, with the presence of redundant information, against which all approaches to RST parsing are nonetheless evaluated.

We proposed to remedy the representational problems by providing an alternate view of the RST data that use dependency structures, a more natural representation that generalizes to other types of discourse structure, as linguistic studies show they correspond to less arbitrary annotation decisions. We argued that dependency structures exhibit a feature that is important to the semantic interpretation of discourse structures; they provide the minimal semantic arguments of discourse relations, while constituent trees constitutes an upper bound. We relied on the notion of headedness in both constituent and dependency views to show how a lot of work on discourse analysis are related. Dependencies also make for more gradual, lenient evaluation, less sensitive to somewhat arbitrary transformations (such as binarization), as was argued in the syntactic community. Because we provide translations from one view to the other, we advocate the use of both evaluation frameworks for comparative purposes.

We compared the predictions of a variety of parsers on the test set of the RST-DT corpus, with evaluation procedures that compute constituency and dependency metrics, to get a better understanding of the differences between approaches. We compared both constituency-based and dependency-based parsers, parsers with a more complex structure that can avail themselves of higher order features, parsers based on learning representations where the complexity lies in the neural architecture and use of the representations, and some simple parsers, which are based on local models with a straightforward decoding technique. We also compared parsers that make use of relatively superficial features and parsers that make use of more sophisticated semantic information. In comparing simple vs. complex parsing strategies, we have seen that among the best in both camps there are at present small differences, even though some approaches perform better on predicting structure and some on predicting relations.

Our empirical comparison of the performance of parsers and the pairwise similarity between their predictions suggest that head-ordered dependency parsing constitutes a viable option for RST discourse parsing, with the promise to adapt more easily to other theories of discourse like SDRT. All of this also suggests that it might be the time for discourse parsing to move away from the very specific dataset that is the RST discourse treebank and aim at better generalizations of discourse models.

Appendix A. Proof of the Equivalence of a Dependency Measure Taking into Account Order and a Measure that Counts Span and Head Accuracy

For simplicity, we look at the case of binary trees. Because any binary projective tree over n elements can be produced from two binary projective trees of less than n elements, we can prove by induction on the construction of projective headed trees of $n + 1$ elements from a projective headed tree of n elements that:

Proposition 3
Suppose T and T' are two binary, headed projective c-trees over a common set of EDUs $\{b_1, ..., b_n\}$ such that $\mathcal{I}(T)$ and $\mathcal{I}(T')$ are d-trees where no node has multiple dependents. Then: $\delta_{SH}(T, T') = \delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$.

The base case for $n = 2$ is easily established by an examination of cases. For the inductive step, assume that T_1 and T_2 are two binary, headed projective c-trees over n EDUs and such that $\mathcal{I}(T_1)$ and $\mathcal{I}(T_2)$ have no nodes with multiple dependents. By construction, we know that T_1 and T_2 are composed from two binary trees over $< n$ EDUs, T_1^1, T_1^2 and T_2^1, T_2^2, respectively, with the same characteristics. By the inductive hypothesis, we know that $\delta_{SH}(T_1^1, T_2^1) = \delta_{uas}(\mathcal{I}(T_1^1), \mathcal{I}(T_2^1))$ and $\delta_{SH}(T_1^2, T_2^2) = \delta_{uas}(\mathcal{I}(T_1^2), \mathcal{I}(T_2^2))$. Note also that $\delta_{SH}(T_1, T_2) = \delta_{SH}(T_1^1, T_2^1) + \delta_{SH}(T_1^2, T_2^2) + 1$, if the head of the T_1 span is distinct from that of the T_2 span. By the inductive hypothesis, then, $\delta_{SH}(T_1, T_2) = \delta_{uas}(\mathcal{I}(T_1), \mathcal{I}(T_2))$.

Now let's bring order into the picture and deal with d-trees that have multiple descendants. To do so, we need to determine a distance measure based on UAS and order. In particular, how do we measure differences in order of operations? In general, because of the projectivity of the RST corpus, we know that there are strong restrictions to how we can apply the different operations; in particular, we know that any set of operations with dependent EDUs to one side of the head must be performed in the linear order of the dependent EDUs. Hence, we can prove by induction over the number of nodes in a d-tree that:

Proposition 4
Let x_0 be the root of a d-tree, with n dependents to the left and m dependents to the right. The number of possible orders for both $n, m > 0$, noted $U_{n,m}$ is:

$$U_{(n-1),m} + U_{n,(m-1)}$$

If either $n = 0$ or $m = 0$, then the number of possible orders is 1: $\forall i,\ U_{i,0} = U_{0,i} = 1$.

Given our map $\mathcal{I}$, Proposition 4 tells us exactly how many different headed RST c-trees we can build from a given d-tree. An immediate consequence of Proposition 4 is that Proposition 3 applies to certain d-trees with multiple dependencies and their headed c-tree equivalents.

Corollary 1
Let T and T' be two, projective headed c-trees over a common set of EDUs $\{b_1, ..., b_n\}$ such that $\mathcal{I}(T)$ and $\mathcal{I}(T')$ are d-trees where any node with n dependents to the left and m dependents to the right is such that either $n = 0$ or $m = 0$. Then: $\delta_{SH}(T, T') = \delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$.

Proposition 4 also gives us some hints as to how to exploit order in a distance definition. If two d-trees have the same head, we can compare the strings of order operations used in each one to generate the tree; an edit distance on these strings can thus inform a OUAS measure. Proposition 3 encodes the fact that one choice may force many others. For example, given a reference ordered d-tree $\{a \leftarrow_2 b, b \rightarrow_1 c, a \rightarrow_3 d\}$, where b is the head, the candidate tree $\{a \leftarrow_1 b, b \rightarrow_2 c, a \rightarrow_3 d\}$ has exactly the same arrows and so hence a perfect UAS score. However, the bracketing shows that in building the candidate tree we made one wrong choice (attaching b first to a), which leads under the translation to a wrong subtree and span (a b instead of b c). If we look at the strings of applications to b, we have $f_{\rightarrow d}(f_{\leftarrow a}(f_{\rightarrow c}(b))$ for the reference tree and $f_{\rightarrow d}(f_{\rightarrow c}(f_{\leftarrow a}(b))$ for the candidate; in counting the wrong choices made, we see that we made one wrong choice of applying $f_{\leftarrow a}$ instead of $f_{\rightarrow c}$; counting such wrong choices would assign the same distance to the candidate tree as would δ_{SH} in this case. That is, $\delta_S H$ and δ_{ouas} agree.

Previously, we have looked at a case where the only difference between two d-trees is a difference in order. What happens when the UAS score for the two trees differs as well? If the two trees have the same head, the simple additive distance combining the edit distance on orders and a UAS score yields something reasonable for simple cases. For instance, consider the two ordered d-trees $T = \{a \leftarrow_2 b, b \rightarrow_1 c, b \rightarrow_3 d\}$ and $T' = \{a \leftarrow_1 b, b \rightarrow_2 d, c \leftarrow d\}$. There is one error in the UAS ($c \leftarrow d$) and two edit distance errors (we have to add the arrow from b to c and that must be numbered first, not the left arrow to a). On the other hand $\delta_{SH}(T, T') = 3$ as well in this case, as there are two incorrect spans and an incorrect head for a span. If the heads of two constituent trees are not the same, we will not be able to compare orders and we set the order distance to 0. Dependency-based measures on ordered d-trees will still capture differences between two such trees, however, because the differences will be reflected in a non-zero UAS distance. The case in (2) is instructive; although the spans are all correct, the heads are wrong and so our δ_{SH} measure agrees with the UAS measure distance of 3.

Proposition 5
Let T and T' be two, projective headed c-trees over a common set of EDUs $\{b_1, \ldots, b_n\}$ such that neither $\mathcal{I}(T)$ or $\mathcal{I}(T')$ share any common head nor do any subtrees of $\mathcal{I}(T)$ or $\mathcal{I}(T')$. Then $\delta_{SH}(T, T') = \delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$.

Given our assumptions, it will follow that $\mathcal{I}(T), \mathcal{I}(T')$ disagree on all edges and so have a maximal δ_{uas} distance of $n - 1$. But it will also follow that $\delta_{SH}(T, T') = n - 1$, as every head in the triples for T' measured will be wrong with respect to the heads in T.

We can now prove by induction that for two trees with a common head, the distances also coincide.

Proposition 6
Let T and T' be two, projective headed c-trees over a common set of EDUs $\{b_1, \ldots, b_n\}$ such that $\mathcal{I}(T)$ and $\mathcal{I}(T')$ share a common head b_k. Then $\delta_{SH}(T, T') = \delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$.

For the base case, consider two d-trees of 3 elements. If they have a common head, then if the common head is a or c, then our result follows from Proposition 1. If the common head is b, then calculation shows that $\delta_{SH}(T, T') = \delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$ in all of the possible configurations. For the inductive step, assume that T_1 and T_2 are two headed projective c-trees over n EDUs with a common head b_k and for binary projective trees over $< n$ EDUs with the same head, the desired result holds. By the same reasoning as in Proposition 3, the result follows. The only difference is that the difference in subtrees between T_1 and T_2 may be reflected not in the UAS but in the order score on the side of the dependency structures.

Summing up our results, Propositions 3, 1, 5, and 6 prove:

Proposition 7
2 Let T and T' be two, projective headed c-trees over a common set of EDUs $\{b_1, \ldots, b_n\}$. Then $\delta_{SH}(T, T') = \delta_{uas}(\mathcal{I}(T), \mathcal{I}(T'))$.

Acknowledgments

We are grateful to the anonymous reviewers for their useful questions and to Pascal Denis and Tim van de Cruys for comments and suggestions throughout the course of this project. This work was supported by STAC, ERC Advanced grant n. 269427.

References

Afantenos, Stergos, Eric Kow, Nicholas Asher, and Jérémy Perret. 2015. Discourse parsing for multi-party chat dialogues. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 928–937, Lisbon.

Asher, Nicholas. 1993. *Reference to Abstract Objects in Discourse*. Kluwer.

Asher, Nicholas, Julie Hunter, Mathieu Morey, Farah Benamara, and Stergos Afantenos. 2016. Discourse structure and dialogue acts in multiparty dialogue: The STAC corpus. In *Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2721–2727, Paris.

Asher, Nicholas and Alex Lascarides. 2003. *Logics of Conversation*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Bhatia, Parminder, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2212–2218, Lisbon.

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Workshop on Speech and Natural Language*, pages 306–311, Pacific Grove, CA.

Braud, Chloé, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual RST discourse parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 292–304, Valencia.

Braud, Chloé, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of RST discourse parsers. In *26th International Conference on Computational Linguistics (COLING)*, pages 1903–1913, Osaka.

Carlson, Lynn, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In Jan van Kuppevelt and Ronnie Smith, editors, *Current Directions in Discourse and Dialogue*. Kluwer Academic Publishers, pages 85–112.

Carroll, J., E. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada.

Carroll, John, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL-99 Workshop on Linguistically Interpreted Corpora*, pages 35–41, Bergen.

Danlos, Laurence. 2004. Discourse dependency structures as constrained dags. In *Proceedings of the SIGDIAL 2004 Workshop, The 5th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 127–135, Cambridge, MA.

duVerle, David and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 665–673, Suntec.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th Conference on Computational linguistics-Volume 1*, pages 340–345, Copenhagen.

Feng, Vanessa Wei and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, MA.

Fernández-González, Daniel and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing.

Gaifman, Haim. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8(3):304–337.

Grosz, B. and C. Sidner. 1986. Attention, intentions and the structure of discourse. *Computational Linguistics*, 12:175–204.

Hayashi, Katsuhiko, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for RST discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136, Los Angeles, CA.

Hernault, Hugo, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Hirao, Tsutomu, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, WA.

Hobbs, J., W. Croft, T. Davies, D. Edwards, and K. Laws. 1987. Commonsense metaphysics and lexical semantics. *Computational Linguistics*, 13(3-4):241–250.

Iruskieta, Mikel, Iria da Cunha, and Maite Taboada. 2015. A qualitative comparison method for rhetorical structures: Identifying different discourse structures in multilingual corpora. *Language Resources and Evaluation*, 49(2):263–309.

Ji, Yangfeng and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, MD.

Ji, Yangfeng and Noah A. Smith. 2017. Neural discourse structure for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 996–1005, Vancouver.

Joty, Shafiq, Giuseppe Carenini, and Raymond T. Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Li, Jiwei, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069, Doha.

Li, Qi, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 362–371, Austin, TX.

Li, Sujian, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25–35, Baltimore, MD.

Louis, Annie, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the SIGDIAL 2010 Conference*, pages 147–156, Tokyo.

Mann, William C. and Sandra A. Thompson. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Marcu, Daniel. 1996. Building up rhetorical structure trees. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Volume 2*, pages 1069–1074, Portland, OR.

Marcu, Daniel. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98, Ann Arbor, MI.

Morey, Mathieu, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on RST discourse parsing? A replication study of recent results on the RST-DT. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1330–1335, Copenhagen.

Muller, Philippe, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING 2012*, pages 1883–1900, Mumbai.

Nigam, Kamal, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering, volume 1*, pages 61–67, Stockholm.

Perret, Jérémy, Stergos Afantenos, Nicholas Asher, and Mathieu Morey. 2016. Integer linear programming for discourse parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 99–109, San Diego, CA.

Prasad, Rashmi, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie L. Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of LREC 2008*, pages 2961–2968, Marrakech.

Reitter, David. 2003. Simple signals for complex rhetorics: On rhetorical analysis with rich-feature support vector models. In *LDV Forum*, volume 18, pages 38–52.

Sagae, Kenji. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 81–84, Stroudsburg, PA.

Sagae, Kenji and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver.

Soricut, Radu and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156, Edmonton.

Surdeanu, Mihai, Thomas Hicks, and Marco A. Valenzuela-Escárcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5, Denver, CO.

van der Vliet, Nynke and Gisela Redeker. 2011. Complex sentences as leaky units in discourse parsing. In *Proceedings of Constraints in Discourse*, pages 1–9, Agay-Saint Raphaël.

Venant, Antoine, Nicholas Asher, Philippe Muller, Pascal Denis, and Stergos Afantenos. 2013. Expressivity and comparison of models of discourse structure. In *Proceedings of the SIGDIAL 2013 Conference*, pages 2–11, Metz.

Wolf, Florian and Edward Gibson. 2006. *Coherence in Natural Language: Data Structures and Applications*. The MIT Press.

Yoshida, Yasuhisa, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1839, Doha.