# Efficient Processing of Underspecified Discourse Representations

**Michaela Regneri**[†] [*]
regneri@coli.uni-sb.de
[*] Saarland University

**Markus Egg**[†]
egg@let.rug.nl
[†] University of Groningen

**Alexander Koller**[§]
a.koller@ed.ac.uk
[§] University of Edinburgh

## Abstract

Underspecification-based algorithms for processing partially disambiguated discourse structure must cope with extremely high numbers of readings. Based on previous work on dominance graphs and weighted tree grammars, we provide the first possibility for computing an underspecified discourse description and a best discourse representation efficiently enough to process even the longest discourses in the RST Discourse Treebank.

## 1 Introduction

Discourse processing has emerged as a highly relevant source of information for applications such as information extraction and automatic summarisation (Taboada and Mann (2006) outline this and further applications). But discourse structures cannot always be described completely, either due to genuine ambiguity (Stede, 2004) or to the limitations of a discourse parser. In either case, only partial information on discourse structure is available. To handle such information, *underspecification* formalisms can be used. Underspecification was originally introduced in computational semantics to model structural ambiguity without disjunctively enumerating the readings, and later applied to discourse parsing (Gardent and Webber, 1998; Schilder, 2002).

However, while the existing algorithms for underspecification processing work well for semantic structures, they were not designed for discourse structures, which can be much larger. Indeed, it has never been shown that underspecified discourse reprentations (UDRs) can be processed efficiently, since the general-purpose implementations are too slow for that task.

In this paper, we present a new way to implement and process discourse underspecification in terms of *regular tree grammars* (RTGs). RTGs are used as an underspecification formalism in semantics (Koller et al., 2008). We show how to compute RTGs for discourse from dominance-based underspecified representations more efficiently (by a typical factor of 100) than before. Furthermore, we show how weighted RTGs can be used to represent constraints and preferences on the discourse structure. Taking all these results together, we show for the first time how the globally optimal discourse representation based on some preference model can be computed efficiently from an UDR.

## 2 Underspecified Discourse Representation

Following annotation schemes like the one of Stede (2004), we model discourse structures by binary trees. Fig. (1b-f) represent the potential structures of (1). We write each elementary discourse unit (EDU) in square brackets.

(1) [$C_1$ I try to read a novel] [$C_2$ if I feel bored] [$C_3$ because the TV programs disappoint me] [$C_4$ but I can't concentrate on anything.]

Underspecification formalisms such as *dominance graphs* (Althaus et al., 2003) can model partial information about such trees; see Fig. (1a) for the underspecified discourse representation (UDR) of (1). These graphs consist of labelled roots and unlabelled holes; the solid edges indicate that a node must be the parent of another, and the dashed edges indicate (transitive) dominance requirements. A *configuration* of a dominance graph is an arrangement of the (labelled) graph nodes into a tree that satisfies all (immediate and transitive) dominance requirements. Subgraphs that are connected by solid edges are called *fragments* and must be tree-shaped.

Using UDRs, discourse parsing can be modularised into three separate steps. First, a discourse parser segments the text and generates an UDR from it. The node labels in the UDR aren't necessarily fully specified (Egg and Redeker, 2007; Schilder,



*Proceedings of ACL-08: HLT, Short Papers (Companion Volume)*, pages 245–248,
Columbus, Ohio, USA, June 2008. 

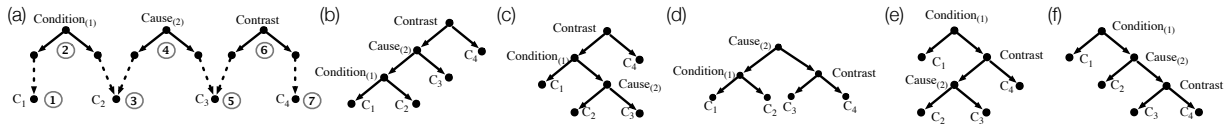


Figure 1: An underspecified discourse structure and its five configurations

2002); here we pretend that they are to simplify the presentation, as nothing in this paper hinges on it. Then weights are added to the UDR that incorporate preferences for discourse structures based on linguistic cues. Finally, the weighted UDR can either be processed directly by other applications, or, if a tree structure is required, we can compute the best configuration. In this paper, we show how an UDR dominance graph can be converted into a regular tree grammar efficiently. This simplifies the specification of weights in Step 2; we also show how to efficiently compute a best tree from a weighted RTG (Step 3). We do not discuss Step 1 in this paper.

The dominance graphs used in discourse underspecification are *constrained chains*. A constrained chain of length $n$ consists of $n$ *upper* fragments with two holes each and $n+1$ *lower* fragments with no holes. There must also be a numbering $1,\ldots,2n+1$ of the fragments such that for every $1 \leq i \leq n$, fragment $2i$ is an upper fragment, fragments $2i-1$ and $2i+1$ are lower fragments, and there are dominance edges from the left hole of $2i$ to the root of $2i-1$ and from the right hole of $2i$ to the root of $2i+1$ (and possibly further dominance edges). These numbers are shown in circles in Fig. (1a). In discourse dominance graphs, upper fragments correspond to discourse relations, and lower fragments correspond to EDUs; the EDUs are ordered according to their appearance in the text, and the upper fragments connect the two text spans to which they are adjacent.

## 3 Underspecified Processing for Discourses

Recently, Koller et al. (2008) showed how to process dominance graphs with *regular tree grammars* (Comon et al., 2007, RTGs). RTGs are a grammar formalism that describes sets of trees using production rules which rewrite non-terminal symbols (NTs) into terms consisting of tree constructors and possibly further NTs. A tree (without NTs) is accepted by the grammar if it can be derived by a sequence of rule applications from a given start symbol. An example RTG is shown in Fig. 2; its start symbol is $\{1;7\}$, and it describes exactly the five trees in

$$
\begin{array}{l|l}
\{1;7\} \to \mathit{Cond}(\{1\},\{3;7\})\ [1] & \{5;7\} \to \mathit{Contr}(\{5\},\{7\})\ [1] \\
\{3;7\} \to \mathit{Contr}(\{3;5\},\{7\})\ [1] & \{3;5\} \to \mathit{Cause}(\{3\},\{5\})\ [1] \\
\{1;7\} \to \mathit{Contr}(\{1;5\},\{7\})\ [1] & \{1;3\} \to \mathit{Cond}(\{1\},\{3\})\ [5] \\
\{1;7\} \to \mathit{Cause}(\{1;3\},\{5;7\})\ [1] & \{1;5\} \to \mathit{Cond}(\{1\},\{3;5\})\ [3] \\
\{1;5\} \to \mathit{Cause}(\{1;3\},\{5\})\ [1] & \{3;7\} \to \mathit{Cause}(\{3\},\{5;7\})\ [1] \\
\{1\} \to C_1\ [1] & \{5\} \to C_3\ [1] \quad \{7\} \to C_4\ [1] \\
\{3\} \to C_2\ [1] &
\end{array}
$$

Figure 2: A wRTG modelling Fig. 1

Fig. (1b-f). For example, Fig. (1e) is derived by expanding the start symbol with the first rule in Fig. 2. This determines that the tree root is labelled with *Condition*; we then derive the left subtree from the NT $\{1\}$ and the right subtree from the NT $\{3;7\}$.

The NTs in the grammar correspond to subgraphs in the dominance graph: The NT $\{1;7\}$ represents the subgraph $\{1,2,3,4,5,6,7\}$ (i.e. the whole graph); the NT $\{1\}$ represents the subgraph containing only the fragment 1; and so forth. The trees that can be derived from each nonterminal correspond exactly to the configurations of the subgraph.

Koller and Thater (2005b) presented an algorithm for generating, from a very general class of dominance graphs, an RTG that describes exactly the same trees. For each subgraph $S$ that is to be the LHS of a rule, the algorithm determines the *free* fragments of $S$, i.e. the fragments that may serve as the root of one of its configurations, by a certain graph algorithm. For every free fragment in $S$ with $n$ holes and a root label $f$, the algorithm generates a new rule of the form $S \to f(S_1,\ldots,S_n)$, where each $S_i$ corresponds to the remaining subgraph under the $i$-th hole. The procedure calls itself recursively on the subgraphs until it reaches singleton subgraphs.

While this algorithm works well with underspecified semantic representations in semantics, it is too slow for the larger discourse graphs, as we will see in Section 5. However, we will now optimise it for the special case of constrained chains. First, we observe that all subgraphs ever visited by the algorithm are connected subchains. A subchain is uniquely identifiable by the positions of the first and last fragment in the left-to-right order of the chain; we can thus read the nonterminal $\{i;j\}$ simply as a pair of integers that identifies the subchain from the $i$-th to the

**Algorithm 1**: GenerateRules($\{i; j\}$,G,C)

```
1  if G contains rules for {i; j} then return
2  if i=j then G.add({ {i; j} → Label(i) } ) else
   /* Loop over upper fragments                    */
3  for k = i+1 to j-1 step 2 do
4      if ¬∃ edge=(s,t) ∈ C s.t. (i ≤ s < k ≤ t ≤ j) ∨ (i ≤ t
       ≤ k < s ≤ j) then
5          lSub ← {i;k-1}, rSub ← {k+1; j}
6          G.add({i; j} → Label(i)(lSub, rSub))
7          GenerateRules(lSub, G, C)
8          GenerateRules(rSub, G, C)
```

$j$-th fragment (rather than an abbreviation for a set of fragments). $i$ and $j$ will generally represent lower fragments. In the grammar in Fig. 2, $\{i\}$ is an abbreviation of $\{i;i\}$.

We can now rephrase the Koller & Thater algorithm in our terms (Algorithm 1). The most important change is that we can now test whether an upper fragment $k$ in a subgraph $\{i; j\}$ is free simply by checking whether there is no dominance edge from some upper fragment $l$ to some upper fragment $r$ such that $i \leq l < k \leq r \leq j$, and no dominance edge from $r$ to $l$ such that $i \leq l \leq k < r \leq j$. For instance, if there was a dominance edge from the right hole of 2 to the root of 6 in Fig. (1a), then 4 and 6 would not be free, but 2 would be; and indeed, all configurations of this graph would have to have 2 as their roots. Hence we can replace the graph algorithm for freeness by a simple comparison of integers. The general structure of the algorithm remains the same as in (Koller and Thater, 2005b): It takes a dominance graph $C$ as its input, and recursively calls itself on pairs $\{i; j\}$ representing subgraphs while adding rules and NTs to an RTG $G$.

## 4 Soft Discourse Constraints

RTGs can be extended to *weighted* regular tree grammars (Knight and Graehl, 2005, wRTGs) by adding numeric weights to the rules. WRTG derivations assign weights to each tree: The weight of a tree is the product of the weights of all rules that were used in its derivation.

Egg and Regneri (2008) motivate the use of wRTGs in discourse processing. They assign rule weights based on corpus-extracted constraints which express the interdependencies between discourse relations and their surrounding tree structure. One such constraint states that the right subtree of a *Condition* node should be of minimal size, which ranks the readings of Fig. 1 (a): (b), (d) > (c) > (e), (f).

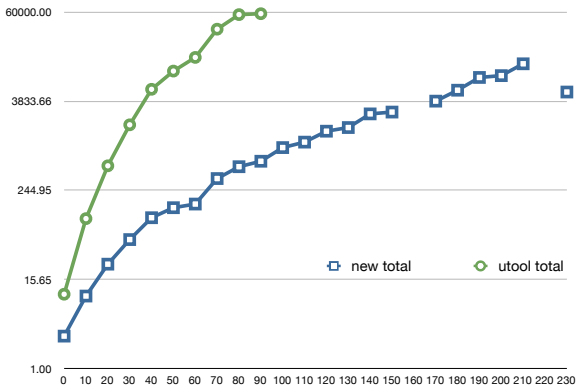


Figure 3: Runtime Comparison

In order to state this constraint in a wRTG, we annotate the grammar in Fig. 2 with the weights shown in brackets. The *Condition* rules get higher weights if the second NT on the RHS represents a smaller subgraph. The grammar assigns the maximum weight of 5 to (b) and (d) (fragment 2 has a leaf as right child), the medium weight 3 to (c) (the right subgraph of fragment 2 contains two EDUs), and the minimum weight 1 to (e) and (f). i.e. it ranks the readings as intended.

Based on our implementation of nonterminals as integer pairs, we can efficiently compute a configuration with maximal weight using a version of Knight and Graehl's (2005) algorithm for computing the best derivation of a wRTG that is specialised to the grammars we use.

## 5 Evaluation

We compare our runtimes with those of Utool (Koller and Thater, 2005a), the fasted known solver for general dominance graphs; it implements the Koller & Thater algorithm. Utool runs very fast for underspecified representations in semantics, but the representations for discourse parsing are considerably larger: The largest underspecified semantic representation found in the Rondane treebank analysed with the English Resource Grammar (Copestake and Flickinger, 2000, ERG) has $4.5 \times 10^{12}$ structural scope readings, but for 59% of the discourses in the RST Discourse Treebank (Carlson et al., 2002, RST-DT), there are more ways of configuring all EDUs into a binary tree than that.

We evaluate the efficiency of our algorithm on 364 texts from the RST-DT, by converting each discourse

into a chain with one lower fragment for each EDU and one upper fragment labelled with each annotated discourse relation. We use our algorithm and Utool to generate the RTG from the chain, assign all soft constraints of Egg and Regneri (2008) to the grammar, and finally compute the best configuration according to this model. The evaluation results are shown in Fig. 3. The horizontal axis shows the chain length (= number of EDUs minus 1), rounded down to multiples of ten; the (logarithmic) vertical axis shows the average runtime in milliseconds for discourses of that length. Both algorithms spend a bit over half the runtime on computing the RTGs.

As the diagram shows, our algorithm is up to 100 times faster than Utool for the same discourses. It is capable of computing the best configuration for every tested discourse – in less than one second for 86% of the texts. Utool exceeded the OS memory limit on 77 discourses, and generally couldn't process any text with more than 100 EDUs. The longest text in the RST-DT has 304 EDUs, so the UDR has about $2.8 \times 10^{178}$ different configurations. Our algorithm computes the best configuration for this UDR in about three minutes.

## 6 Conclusion

We presented the first solver for underspecified discourse representations that is efficient enough to compute the globally best configurations of every discourse in the RST discourse treebank, by exploiting the fact that UDRs are very large but obey very strong structural restrictions. Our solver converts a dominance graph into an RTG, adds weights to the RTG to represent discourse constraints, and then computes the globally optimal configuration.

It takes about three minutes to compute a best configuration with a given probability model for the longest discourse in the treebank, out of $10^{178}$ possible configurations. For comparison, an algorithm that enumerates a billion configurations per second to find the best one could have inspected only about $10^{26}$ within the estimated age of the universe. So our algorithm is useful and necessary to process real-world underspecified discourse representations.

We have thus demonstrated that discourse processing based on underspecification is computationally feasible. Nothing in our algorithm hinges on using RST in particular; it is compatible with any approach that uses binary trees. In future research, it would be interesting to complete our system into a full-blown discourse parser by adding a module that computes an UDR for a given text, and evaluate whether its ability to delay decisions about discourse structure would improve accuracy.

## References


E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48:194–219.

L. Carlson, D. Marcu, and M. E. Okurowski. 2002. RST Discourse Treebank. LDC.

H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree Automata Techniques and Applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. Release 12-10-2007.

A. Copestake and D. Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Conference on Language Resources and Evaluation*.

M. Egg and G. Redeker. 2007. Underspecified discourse representation. In A. Benz and P. Kühnlein, editors, *Constraints in Discourse*, Amsterdam. Benjamins.

M. Egg and M. Regneri. 2008. Underspecified Modelling of Complex Discourse Constraints. Submitted.

C. Gardent and B. Webber. 1998. Describing Discourse Semantics. In *Proceedings of the 4th TAG+ Workshop*, University of Pennsylvania, Philadelphia.

K. Knight and J. Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Computational linguistics and intelligent text processing*, pages 1–24. Springer.

A. Koller and S. Thater. 2005a. Efficient solving and exploration of scope ambiguities. Proceedings of the ACL-05 Demo Session.

A. Koller and S. Thater. 2005b. The evolution of dominance constraint solvers. In *Proceedings of the ACL-05 Workshop on Software*, Ann Arbor.

A. Koller, M. Regneri, and S. Thater. 2008. Regular tree grammars as a formalism for scope underspecification. In *Proceedings of ACL-08: HLT*.

F. Schilder. 2002. Robust discourse parsing via discourse markers, topicality and position. *Natural Language Engineering*, 8:235–255.

M. Stede. 2004. The Potsdam Commentary Corpus. In B. Webber and D. Byron, editors, *ACL-04 Workshop on Discourse Annotation*.

M. Taboada and W. Mann. 2006. Applications of Rhetorical Structure Theory. *Discourse Studies*, 8:567–588.