# Regular tree grammars as a formalism for scope underspecification

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

**Michaela Regneri**[† §]
regneri@coli.uni-sb.de
[†] University of Groningen

**Stefan Thater**[§]
stth@coli.uni-sb.de
[§] Saarland University

## Abstract

We propose the use of regular tree grammars (RTGs) as a formalism for the underspecified processing of scope ambiguities. By applying standard results on RTGs, we obtain a novel algorithm for eliminating equivalent readings and the first efficient algorithm for computing the best reading of a scope ambiguity. We also show how to derive RTGs from more traditional underspecified descriptions.

## 1 Introduction

Underspecification (Reyle, 1993; Copestake et al., 2005; Bos, 1996; Egg et al., 2001) has become the standard approach to dealing with scope ambiguity in large-scale hand-written grammars (see e.g. Copestake and Flickinger (2000)). The key idea behind underspecification is that the parser avoids computing all scope readings. Instead, it computes a single compact *underspecified description* for each parse. One can then strengthen the underspecified description to efficiently eliminate subsets of readings that were not intended in the given context (Koller and Niehren, 2000; Koller and Thater, 2006); so when the individual readings are eventually computed, the number of remaining readings is much smaller and much closer to the actual perceived ambiguity of the sentence.

In the past few years, a "standard model" of scope underspecification has emerged: A range of formalisms from Underspecified DRT (Reyle, 1993) to dominance graphs (Althaus et al., 2003) have offered mechanisms to specify the "semantic material" of which the semantic representations are built up, plus dominance or outscoping relations between these building blocks. This has been a very successful approach, but recent algorithms for eliminating subsets of readings have pushed the expressive power of these formalisms to their limits; for instance, Koller and Thater (2006) speculate that further improvements over their (incomplete) redundancy elimination algorithm require a more expressive formalism than dominance graphs. On the theoretical side, Ebert (2005) has shown that none of the major underspecification formalisms are *expressively complete*, i.e. supports the description of an arbitrary subset of readings. Furthermore, the somewhat implicit nature of dominance-based descriptions makes it difficult to systematically associate readings with probabilities or costs and then compute a best reading.

In this paper, we address both of these shortcomings by proposing *regular tree grammars (RTGs)* as a novel underspecification formalism. Regular tree grammars (Comon et al., 2007) are a standard approach for specifying sets of trees in theoretical computer science, and are closely related to regular tree transducers as used e.g. in recent work on statistical MT (Knight and Graehl, 2005) and grammar formalisms (Shieber, 2006). We show that the "dominance charts" proposed by Koller and Thater (2005b) can be naturally seen as regular tree grammars; using their algorithm, classical underspecified descriptions (dominance graphs) can be translated into RTGs that describe the same sets of readings. However, RTGs are trivially expressively complete because every finite tree language is also regular. We exploit this increase in expressive power in presenting a novel redundancy elimination algorithm that is simpler and more powerful than the one by Koller and Thater (2006); in our algorithm, redundancy elimination amounts to intersection of regular tree languages. Furthermore, we show how to define a PCFG-style cost model on RTGs and compute best readings of deterministic RTGs efficiently, and illustrate this model on a machine learning based model



*Proceedings of ACL-08: HLT*, pages 218–226,
Columbus, Ohio, USA, June 2008. 

of scope preferences (Higgins and Sadock, 2003). To our knowledge, this is the first efficient algorithm for computing best readings of a scope ambiguity in the literature.

The paper is structured as follows. In Section 2, we will first sketch the existing standard approach to underspecification. We will then define regular tree grammars and show how to see them as an underspecification formalism in Section 3. We will present the new redundancy elimination algorithm, based on language intersection, in Section 4, and show how to equip RTGs with weights and compute best readings in Section 5. We conclude in Section 6.

## 2 Underspecification

The key idea behind scope underspecification is to describe all readings of an ambiguous expression with a single, compact underspecified representation (USR). This simplifies semantics construction, and current algorithms (Koller and Thater, 2005a) support the efficient enumeration of readings from an USR when it is necessary. Furthermore, it is possible to perform certain semantic processing tasks such as eliminating redundant readings (see Section 4) directly on the level of underspecified representations without explicitly enumerating individual readings.

Under the "standard model" of scope underspecification, readings are considered as formulas or trees. USRs specify the "semantic material" common to all readings, plus dominance or outscopes relations between these building blocks. In this paper, we consider dominance graphs (Egg et al., 2001; Althaus et al., 2003) as one representative of this class. An example dominance graph is shown on the left of Fig. 1. It represents the five readings of the sentence "a representative of a company saw every sample." The (directed, labelled) graph consists of seven subtrees, or *fragments*, plus *dominance edges* relating nodes of these fragments. Each reading is encoded as one *configuration* of the dominance graph, which can be obtained by "plugging" the tree fragments into each other, in a way that respects the dominance edges: The source node of each dominance edge must dominate (i.e., be an ancestor of) the target node in each configuration. The trees in Fig. 1a–e are the five configurations of the example graph.

An important class of dominance graphs are *hypernormally connected* dominance graphs, or *dominance nets* (Niehren and Thater, 2003). The precise definition of dominance nets is not important here, but note that virtually all underspecified descriptions that are produced by current grammars are nets (Flickinger et al., 2005). For the rest of the paper, we restrict ourselves to dominance graphs that are hypernormally connected.

## 3 Regular tree grammars

We will now recall the definition of regular tree grammars and show how they can be used as an underspecification formalism.

### 3.1 Definition

Let $\Sigma$ be an alphabet, or signature, of tree constructors $\{f, g, a, \ldots\}$, each of which is equipped with an arity $\mathsf{ar}(f) \geq 0$. A *finite constructor tree* $t$ is a finite tree in which each node is labelled with a symbol of $\Sigma$, and the number of children of the node is exactly the arity of this symbol. For instance, the configurations in Fig. 1a-e are finite constructor trees over the signature $\{a_x|2, a_y|2, comp_z|0, \ldots\}$. Finite constructor trees can be seen as ground terms over $\Sigma$ that respect the arities. We write $T(\Sigma)$ for the finite constructor trees over $\Sigma$.

A *regular tree grammar* (RTG) is a 4-tuple $G = (S, N, \Sigma, R)$ consisting of a *nonterminal alphabet* $N$, a *terminal alphabet* $\Sigma$, a *start symbol* $S \in N$, and a finite set of *production rules* $R$ of the form $A \to \beta$, where $A \in N$ and $\beta \in T(\Sigma \cup N)$; the nonterminals count as zero-place constructors. Two finite constructor trees $t, t' \in T(\Sigma \cup N)$ stand in the *derivation relation*, $t \to_G t'$, if $t'$ can be built from $t$ by replacing an occurrence of some nonterminal $A$ by the tree on the right-hand side of some production for $A$. The *language generated by* $G$, $L(G)$, is the set $\{t \in T(\Sigma) \mid S \to_G^* t\}$, i.e. all terms of terminal symbols that can be derived from the start symbol by a sequence of rule applications. Note that $L(G)$ is a possibly infinite language of finite trees. As usual, we write $A \to t_1 \mid \ldots \mid t_n$ as shorthand for the $n$ production rules $A \to t_i$ $(1 \leq i \leq n)$. See Comon et al. (2007) for more details.

The languages that can be accepted by regular tree grammars are called *regular tree languages (RTLs)*, and regular tree grammars are equivalent to *regular*

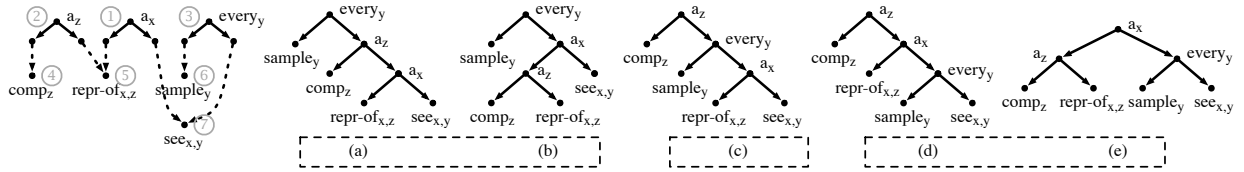


Figure 1: A dominance graph (left) and its five configurations.

*tree automata*, which are defined essentially like the well-known regular string automata, except that they assign states to the nodes in a tree rather than the positions in a string. Tree automata are related to tree transducers as used e.g. in statistical machine translation (Knight and Graehl, 2005) exactly like finite-state string automata are related to finite-state string transducers, i.e. they use identical mechanisms to accept rather than transduce languages. Many theoretical results carry over from regular string languages to regular tree languages; for instance, membership of a tree in a RTL can be decided in linear time, RTLs are closed under intersection, union, and complement, and so forth.

### 3.2 Regular tree grammars in underspecification

We can now use regular tree grammars in underspecification by representing the semantic representations as trees and taking an RTG $G$ as an underspecified description of the trees in $L(G)$. For example, the five configurations in Fig. 1 can be represented as the tree language accepted by the following grammar with start symbol $S$.

$$
\begin{array}{rcll}
S & \to & a_x(A_1,A_2) \mid a_z(B_1,A_3) \mid every_y(B_3,A_4) & \\
A_1 & \to & a_z(B_1,B_2) & \\
A_2 & \to & every_y(B_3,B_4) & \\
A_3 & \to & a_x(B_2,A_2) \mid every_y(B_3,A_5) & \\
A_4 & \to & a_x(A_1,B_4) \mid a_z(B_1,A_5) & \\
A_5 & \to & a_x(B_2,B_4) & \\
B_1 & \to & comp_z & B_2 \to repr\text{-}of_{x,z} \\
B_3 & \to & sample_y & B_4 \to see_{x,y}
\end{array}
$$

More generally, every finite set of trees can be written as the tree language accepted by a non-recursive regular tree grammar such as this. This grammar can be much smaller than the set of trees, because nonterminal symbols (which stand for sets of possibly many subtrees) can be used on the right-hand sides of multiple rules. Thus an RTG is a compact representation of a set of trees in the same way that a parse chart is a compact representation of the set of parse trees of a context-free string grammar. Note that each tree can be enumerated from the RTG in linear time.

### 3.3 From dominance graphs to tree grammars

Furthermore, regular tree grammars can be systematically computed from more traditional underspecified descriptions. Koller and Thater (2005b) demonstrate how to compute a *dominance chart* from a dominance graph $D$ by tabulating how a subgraph can be decomposed into smaller subgraphs by removing what they call a "free fragment". If $D$ is hypernormally connected, this chart can be read as a regular tree grammar whose nonterminal symbols are subgraphs of the dominance graph, and whose terminal symbols are names of fragments. For the example graph in Fig. 1, it looks as follows.

$$
\begin{array}{rcl}
\{1,2,3,4,5,6,7\} & \to & 1(\{2,4,5\},\{3,6,7\}) \\
\{1,2,3,4,5,6,7\} & \to & 2(\{4\},\{1,3,5,6,7\}) \\
\{1,2,3,4,5,6,7\} & \to & 3(\{6\},\{1,2,4,5,7\}) \\
\{1,3,5,6,7\} & \to & 1(\{5\},\{3,6,7\}) \mid 3(\{6\},\{1,5,7\}) \\
\{1,2,4,5,7\} & \to & 1(\{2,4,5\},\{7\}) \mid 2(\{4\},\{1,5,7\}) \\
\{1,5,7\} & \to & 1(\{5\},\{7\})
\end{array}
$$

$$
\begin{array}{llll}
\{2,4,5\} \to 2(\{4\},\{5\}) & \{4\} \to 4 & \{6\} \to 6 \\
\{3,6,7\} \to 3(\{6\},\{7\}) & \{5\} \to 5 & \{7\} \to 7
\end{array}
$$

This grammar accepts, again, five different trees, whose labels are the node names of the dominance graph, for instance $1(2(4,5),3(6,7))$. If $f : \Sigma \to \Sigma'$ is a *relabelling function* from one terminal alphabet to another, we can write $f(G)$ for the grammar $(S,N,\Sigma',R')$, where $R' = \{A \to f(a)(B_1,\ldots,B_n) \mid A \to a(B_1,\ldots,B_n) \in R\}$. Now if we choose $f$ to be the labelling function of $D$ (which maps node names to node labels) and $G$ is the chart of $D$, then $L(f(G))$ will be the set of configurations of $D$. The grammar in Section 3.2 is simply $f(G)$ for the chart above (up to consistent renaming of nonterminals).

In the worst case, the dominance chart of a dominance graph with $n$ fragments has $O(2^n)$ production rules (Koller and Thater, 2005b), i.e. charts may be exponential in size; but note that this is still an

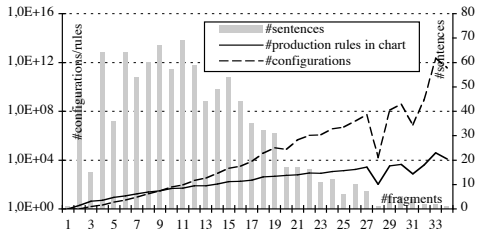


Figure 2: Chart sizes in the Rondane corpus.

improvement over the $n!$ configurations that these worst-case examples have. In practice, RTGs that are computed by converting the USR computed by a grammar remain compact: Fig. 2 compares the average number of configurations and the average number of RTG production rules for USRs of increasing sizes in the Rondane treebank (see Sect. 4.3); the bars represent the number of sentences for USRs of a certain size. Even for the most ambiguous sentence, which has about $4.5 \times 10^{12}$ scope readings, the dominance chart has only about 75 000 rules, and it takes only 15 seconds on a modern consumer PC (Intel Core 2 Duo at 2 GHz) to compute the grammar from the graph. Computing the charts for all 999 MRS-nets in the treebank takes about 45 seconds.

## 4 Expressive completeness and redundancy elimination

Because every finite tree language is regular, RTGs constitute an *expressively complete* underspecification formalism in the sense of Ebert (2005): They can represent arbitrary subsets of the original set of readings. Ebert shows that the classical dominance-based underspecification formalisms, such as MRS, Hole Semantics, and dominance graphs, are all expressively incomplete, which Koller and Thater (2006) speculate might be a practical problem for algorithms that strengthen USRs to remove unwanted readings. We will now show how both the expressive completeness and the availability of standard constructions for RTGs can be exploited to get an improved redundancy elimination algorithm.

### 4.1 Redundancy elimination

Redundancy elimination (Vestre, 1991; Chaves, 2003; Koller and Thater, 2006) is the problem of deriving from an USR $U$ another USR $U'$, such that the readings of $U'$ are a proper subset of the readings of $U$, but every reading in $U$ is semantically equivalent to some reading in $U'$. For instance, the following sentence from the Rondane treebank is analyzed as having six quantifiers and 480 readings by the ERG grammar; these readings fall into just two semantic equivalence classes, characterized by the relative scope of "the lee of" and "a small hillside". A redundancy elimination would therefore ideally reduce the underspecified description to one that has only two readings (one for each class).

(1) We quickly put up the tents in the lee of a small hillside and cook for the first time in the open. (Rondane 892)

Koller and Thater (2006) define semantic equivalence in terms of a rewrite system that specifies under what conditions two quantifiers may exchange their positions without changing the meaning of the semantic representation. For example, if we assume the following rewrite system (with just a single rule), the five configurations in Fig. 1a-e fall into three equivalence classes – indicated by the dotted boxes around the names a-e – because two pairs of readings can be rewritten into each other.

(2) $a_x(a_z(P,Q),R) \rightarrow a_z(P,a_x(Q,R))$

Based on this definition, Koller and Thater (2006) present an algorithm (henceforth, KT06) that deletes rules from a dominance chart and thus removes subsets of readings from the USR. The KT06 algorithm is fast and quite effective in practice. However, it essentially predicts for each production rule of a dominance chart whether each configuration that can be built with this rule is equivalent to a configuration that can be built with some other production for the same subgraph, and is therefore rather complex.

### 4.2 Redundancy elimination as language intersection

We now define a new algorithm for redundancy elimination. It is based on the intersection of regular tree languages, and will be much simpler and more powerful than KT06.

Let $G = (S,N,\Sigma,R)$ be an RTG with a linear order on the terminals $\Sigma$; for ease of presentation, we assume $\Sigma \subseteq \mathbb{N}$. Furthermore, let $f : \Sigma \rightarrow \Sigma'$ be a relabelling function into the signature $\Sigma'$ of the rewrite

system. For example, $G$ could be the dominance chart of some dominance graph $D$, and $f$ could be the labelling function of $D$.

We can then define a tree language $L_F$ as follows: $L_F$ contains all trees over $\Sigma$ that do not contain a subtree of the form $q_1(x_1,\ldots,x_{i-1},q_2(\ldots),x_{i+1},\ldots,x_k)$ where $q_1 > q_2$ and the rewrite system contains a rule that has $f(q_1)(X_1,\ldots,X_{i-1},f(q_2)(\ldots),X_{i+1},\ldots,X_k)$ on the left or right hand side. $L_F$ is a regular tree language, and can be accepted by a regular tree grammar $G_F$ with $O(n)$ nonterminals and $O(n^2)$ rules, where $n = |\Sigma'|$. A filter grammar for Fig. 1 looks as follows:

$$\begin{array}{rl} S & \to 1(S,S) \mid 2(S,Q_1) \mid 3(S,S) \mid 4 \mid \ldots \mid 7 \\ Q_1 & \to 2(S,Q_1) \mid 3(S,S) \mid 4 \mid \ldots \mid 7 \end{array}$$

This grammar accepts all trees over $\Sigma$ except ones in which a node with label 2 is the parent of a node with label 1, because such trees correspond to configurations in which a node with label $a_z$ is the parent of a node with label $a_x$, $a_z$ and $a_x$ are permutable, and $2 > 1$. In particular, it will accept the configurations (b), (c), and (e) in Fig. 1, but not (a) or (d).

Since regular tree languages are closed under intersection, we can compute a grammar $G'$ such that $L(G') = L(G) \cap L_F$. This grammar has $O(nk)$ nonterminals and $O(n^2k)$ productions, where $k$ is the number of production rules in $G$, and can be computed in time $O(n^2k)$. The relabelled grammar $f(G')$ accepts all trees in which adjacent occurrences of permutable quantifiers are in a canonical order (sorted from lowest to highest node name). For example, the grammar $G'$ for the example looks as follows; note that the nonterminal alphabet of $G'$ is the product of the nonterminal alphabets of $G$ and $G_F$.

$$\begin{array}{rcl} \{1,2,3,4,5,6,7\}_S & \to & 1(\{2,4,5\}_S,\{3,6,7\}_S) \\ \{1,2,3,4,5,6,7\}_S & \to & 2(\{4\}_S,\{1,3,5,6,7\}_{Q_1}) \\ \{1,2,3,4,5,6,7\}_S & \to & 3(\{6\}_S,\{1,2,4,5,7\}_S) \\ \{1,3,5,6,7\}_{Q_1} & \to & 3(\{6\}_S,\{1,5,7\}_S) \\ \{1,2,4,5,7\}_S & \to & 1(\{2,4,5\}_S,\{7\}_S) \\ \{1,2,4,5,7\}_S & \to & 2(\{4\}_S,\{1,5,7\}_{Q_1}) \end{array}$$

$$\begin{array}{lll} \{2,4,5\}_S \to 2(\{4\}_S,\{5\}_{Q_1}) & & \{4\}_S \to 4 \\ \{3,6,7\}_S \to 3(\{6\}_S,\{7\}_S) & & \{5\}_S \to 5 \\ \{1,5,7\}_S \to 1(\{5\}_S,\{7\}_S) & & \{5\}_{Q_1} \to 5 \\ \{6\}_S \to 6 & & \{7\}_S \to 7 \end{array}$$

Significantly, the grammar contains no productions for $\{1,3,5,6,7\}_{Q_1}$ with terminal symbol 1, and no production for $\{1,5,7\}_{Q_1}$. This reduces the tree language accepted by $f(G')$ to just the configurations (b), (c), and (e) in Fig. 1, i.e. exactly one representative of every equivalence class. Notice that there are two different nonterminals, $\{5\}_{Q_1}$ and $\{5\}_S$, corresponding to the subgraph $\{5\}$, so the intersected RTG is not a dominance chart any more. As we will see below, this increased expressivity increases the power of the redundancy elimination algorithm.

### 4.3 Evaluation

The algorithm presented here is not only more transparent than KT06, but also more powerful; for example, it will reduce the graph in Fig. 4 of Koller and Thater (2006) completely, whereas KT06 won't.

To measure the extent to which the new algorithm improves upon KT06, we compare both algorithms on the USRs in the Rondane treebank (version of January 2006). The Rondane treebank is a "Redwoods style" treebank (Oepen et al., 2002) containing MRS-based underspecified representations for sentences from the tourism domain, and is distributed together with the English Resource Grammar (ERG) (Copestake and Flickinger, 2000).

The treebank contains 999 MRS-nets, which we translate automatically into dominance graphs and further into RTGs; the median number of scope readings per sentence is 56. For our experiment, we consider all 950 MRS-nets with less than 650 000 configurations. We use a slightly weaker version of the rewrite system that Koller and Thater (2006) used in their evaluation.

It turns out that the median number of equivalence classes, computed by pairwise comparison of all configurations, is 8. The median number of configurations that remain after running our algorithm is also 8. By contrast, the median number after running KT06 is 11. For a more fine-grained comparison, Fig. 3 shows the percentage of USRs for which the two algorithms achieve complete reduction, i.e. retain only one reading per equivalence class. In the diagram, we have grouped USRs according to the natural logarithm of their numbers of configurations, and report the percentage of USRs in this group on which the algorithms were complete. The new algorithm dramatically outperforms KT06: In total, it reduces 96% of all USRs completely, whereas KT06 was complete only for 40%. This increase in completeness is partially due to the new algorithm's ability to use non-chart RTGs: For 28% of the sentences,

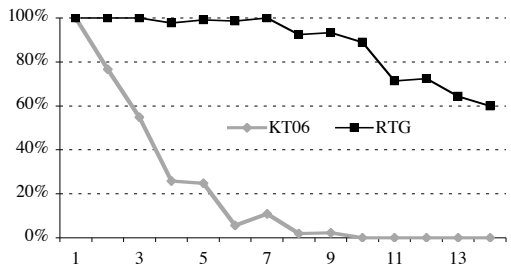

Figure 3: Percentage of USRs in Rondane for which the algorithms achieve complete reduction.

it computes RTGs that are not dominance charts. KT06 was only able to reduce 5 of these 263 graphs completely.

The algorithm needs 25 seconds to run for the entire corpus (old algorithm: 17 seconds), and it would take 50 (38) more seconds to run on the 49 large USRs that we exclude from the experiment. By contrast, it takes about 7 hours to compute the equivalence classes by pairwise comparison, and it would take an estimated several billion years to compute the equivalence classes of the excluded USRs. In short, the redundancy elimination algorithm presented here achieves nearly complete reduction at a tiny fraction of the runtime, and makes a useful task that was completely infeasible before possible.

### 4.4 Compactness

Finally, let us briefly consider the ramifications of expressive completeness on efficiency. Ebert (2005) proves that no expressively complete underspecification formalism can be *compact*, i.e. in the worst case, the USR of a set of readings become exponentially large in the number of scope-bearing operators. In the case of RTGs, this worst case is achieved by grammars of the form $S \to t_1 \mid \ldots \mid t_n$, where $t_1, \ldots, t_n$ are the trees we want to describe. This grammar is as big as the number of readings, i.e. worst-case exponential in the number $n$ of scope-bearing operators, and essentially amounts to a meta-level disjunction over the readings.

Ebert takes the incompatibility between compactness and expressive completeness as a fundamental problem for underspecification. We don't see things quite as bleakly. Expressions of natural language itself are (extremely underspecified) descriptions of sets of semantic representations, and so Ebert's argument applies to NL expressions as well. This means that describing a given set of readings may require an exponentially long discourse. Ebert's definition of compactness may be too harsh: An USR, although exponential-size in the number of quantifiers, may still be polynomial-size in the length of the discourse in the worst case.

Nevertheless, the tradeoff between compactness and expressive power is important for the design of underspecification formalisms, and RTGs offer a unique answer. They are expressively complete; but as we have seen in Fig. 2, the RTGs that are derived by semantic construction are compact, and even intersecting them with filter grammars for redundancy elimination only blows up their sizes by a factor of $O(n^2)$. As we add more and more information to an RTG to reduce the set of readings, ultimately to those readings that were meant in the actual context of the utterance, the grammar will become less and less compact; but this trend is counterbalanced by the overall reduction in the number of readings. For the USRs in Rondane, the intersected RTGs are, on average, 6% smaller than the original charts. Only 30% are larger than the charts, by a maximal factor of 3.66. Therefore we believe that the theoretical non-compactness should not be a major problem in a well-designed practical system.

## 5 Computing best configurations

A second advantage of using RTGs as an underspecification formalism is that we can apply existing algorithms for computing the best derivations of *weighted* regular tree grammars to compute best (that is, cheapest or most probable) configurations. This gives us the first efficient algorithm for computing the preferred reading of a scope ambiguity.

We define weighted dominance graphs and weighted tree grammars, show how to translate the former into the latter and discuss an example.

### 5.1 Weighted dominance graphs

A *weighted* dominance graph $D = (V, E_T \uplus E_D \uplus W_D \uplus W_I)$ is a dominance graph with two new types of edges – *soft dominance edges*, $W_D$, and *soft disjointness edges*, $W_I$ –, each of which is equipped with a numeric weight. Soft dominance and disjointness edges provide a mechanism for assigning weights to configurations; a soft dominance edge ex-

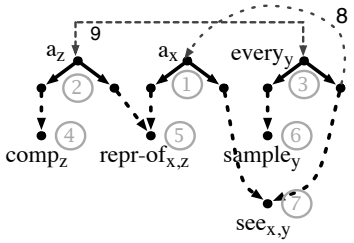


Figure 4: The graph of Fig. 1 with soft constraints

presses a preference that two nodes dominate each other in a configuration, whereas a soft disjointness edge expresses a preference that two nodes are *disjoint*, i.e. neither dominates the other.

We take the *hard backbone* of $D$ to be the ordinary dominance graph $B(D) = (V, E_T \uplus E_D)$ obtained by removing all soft edges. The set of *configurations* of a weighted graph $D$ is the set of configurations of its hard backbone. For each configuration $t$ of $D$, we define the *weight* $c(t)$ to be the product of the weights of all soft dominance and disjointness edges that are satisfied in $t$. We can then ask for configurations of maximal weight.

Weighted dominance graphs can be used to encode the standard models of scope preferences (Pafel, 1997; Higgins and Sadock, 2003). For example, Higgins and Sadock (2003) present a machine learning approach for determining pairwise preferences as to whether a quantifier $Q_1$ dominates another quantifier $Q_2$, $Q_2$ dominates $Q_1$, or neither (i.e. they are disjoint). We can represent these numbers as the weights of soft dominance and disjointness edges. An example (with artificial weights) is shown in Fig. 4; we draw the soft dominance edges as curved dotted arrows and the soft disjointness edges as as angled double-headed arrows. Each soft edge is annotated with its weight. The hard backbone of this dominance graph is our example graph from Fig. 1, so it has the same five configurations. The weighted graph assigns a weight of 8 to configuration (a), a weight of 1 to (d), and a weight of 9 to (e); this is also the configuration of maximum weight.

### 5.2 Weighted tree grammars

In order to compute the maximal-weight configuration of a weighted dominance graph, we will first translate it into a *weighted regular tree grammar*. A weighted regular tree grammar (wRTG) (Graehl and Knight, 2004) is a 5-tuple $G = (S, N, \Sigma, R, c)$ such that $G' = (S, N, \Sigma, R)$ is a regular tree grammar and $c : R \to \mathbb{R}$ is a function that assigns each production rule a weight. $G$ accepts the same language of trees as $G'$. It assigns each derivation a cost equal to the product of the costs of the production rules used in this derivation, and it assigns each tree in the language a cost equal to the sum of the costs of its derivations. Thus wRTGs define weights in a way that is extremely similar to PCFGs, except that we don't require any weights to sum to one.

Given a weighted, hypernormally connected dominance graph $D$, we can extend the chart of $B(D)$ to a wRTG by assigning rule weights as follows: The weight of a rule $D_0 \to i(D_1, \ldots, D_n)$ is the product over the weights of all soft dominance and disjointness edges that are established by this rule. We say that a rule establishes a soft dominance edge from $u$ to $v$ if $u = i$ and $v$ is in one of the subgraphs $D_1, \ldots, D_n$; we say that it establishes a soft disjointness edge between $u$ and $v$ if $u$ and $v$ are in different subgraphs $D_j$ and $D_k$ ($j \neq k$). It can be shown that the weight this grammar assigns to each derivation is equal to the weight that the original dominance graph assigns to the corresponding configuration.

If we apply this construction to the example graph in Fig. 4, we obtain the following wRTG:

$$
\begin{array}{rrll}
 & \{1,...,7\} & \to & a_x(\{2,4,5\},\{3,6,7\}) \quad [9] \\
 & \{1,...,7\} & \to & a_z(\{4\},\{1,3,5,6,7\}) \quad [1] \\
 & \{1,...,7\} & \to & every_y(\{6\},\{1,2,4,5,7\}) \quad [8] \\
 & \{2,4,5\} & \to & a_z(\{4\},\{5\}) \quad [1] \\
 & \{3,6,7\} & \to & every_y(\{6\},\{7\}) \quad [1] \\
 & \{1,3,5,6,7\} & \to & a_x(\{5\},\{3,6,7\}) \quad [1] \\
 & \{1,3,5,6,7\} & \to & every_y(\{6\},\{1,5,7\}) \quad [8] \\
 & \{1,2,4,5,7\} & \to & a_x(\{2,4,5\},\{7\}) \quad [1] \\
 & \{1,2,4,5,7\} & \to & a_z(\{4\},\{1,5,7\}) \quad [1] \\
 & \{1,5,7\} & \to & a_x(\{5\},\{7\}) \quad [1] \\
\{4\} \to & comp_z & [1] & \{5\} \to repr{-}of_{x,z} \quad [1] \\
\{6\} \to & sample_y & [1] & \{7\} \to see_{x,y} \quad [1]
\end{array}
$$

For example, picking "$a_z$" as the root of a configuration (Fig. 1 (c), (d)) of the entire graph has a weight of 1, because this rule establishes no soft edges. On the other hand, choosing "$a_x$" as the root has a weight of 9, because this establishes the soft disjointness edge (and in fact, leads to the derivation of the maximum-weight configuration in Fig. 1 (e)).

### 5.3 Computing the best configuration

The problem of computing the best configuration of a weighted dominance graph – or equivalently, the

best derivation of a weighted tree grammar – can now be solved by standard algorithms for wRTGs. For example, Knight and Graehl (2005) present an algorithm to extract the best derivation of a wRTG in time $O(t + n \log n)$ where $n$ is the number of nonterminals and $t$ is the number of rules. In practice, we can extract the best reading of the most ambiguous sentence in the Rondane treebank ($4.5 \times 10^{12}$ readings, 75 000 grammar rules) with random soft edges in about a second.

However, notice that this is not the same problem as computing the best *tree* in the language accepted by a wRTG, as trees may have multiple derivations. The problem of computing the best tree is NP-complete (Sima'an, 1996). However, if the weighted regular tree automaton corresponding to the wRTG is deterministic, every tree has only one derivation, and thus computing best trees becomes easy again. The tree automata for dominance charts are always deterministic, and the automata for RTGs as in Section 3.2 (whose terminals correspond to the graph's node labels) are also typically deterministic if the variable names are part of the quantifier node labels. Furthermore, there are algorithms for determinizing weighted tree automata (Borchardt and Vogler, 2003; May and Knight, 2006), which could be applied as preprocessing steps for wRTGs.

## 6 Conclusion

In this paper, we have shown how regular tree grammars can be used as a formalism for scope underspecification, and have exploited the power of this view in a novel, simpler, and more complete algorithm for redundancy elimination and the first efficient algorithm for computing the best reading of a scope ambiguity. In both cases, we have adapted standard algorithms for RTGs, which illustrates the usefulness of using such a well-understood formalism. In the worst case, the RTG for a scope ambiguity is exponential in the number of scope bearers in the sentence; this is a necessary consequence of their expressive completeness. However, those RTGs that are computed by semantic construction and redundancy elimination remain compact.

Rather than showing how to do semantic construction for RTGs, we have presented an algorithm that computes RTGs from more standard underspecification formalisms. We see RTGs as an "underspecification assembly language" – they support efficient and useful algorithms, but direct semantic construction may be inconvenient, and RTGs will rather be obtained by "compiling" higher-level underspecified representations such as dominance graphs or MRS.

This perspective also allows us to establish a connection to approaches to semantic construction which use chart-based packing methods rather than dominance-based underspecification to manage scope ambiguities. For instance, both Combinatory Categorial Grammars (Steedman, 2000) and synchronous grammars (Nesson and Shieber, 2006) represent syntactic and semantic ambiguity as part of the same parse chart. These parse charts can be seen as regular tree grammars that accept the language of parse trees, and conceivably an RTG that describes only the semantic and not the syntactic ambiguity could be automatically extracted. We could thus reconcile these completely separate approaches to semantic construction within the same formal framework, and RTG-based algorithms (e.g., for redundancy elimination) would apply equally to dominance-based and chart-based approaches. Indeed, for one particular grammar formalism it has even been shown that the parse chart contains an isomorphic image of a dominance chart (Koller and Rambow, 2007).

Finally, we have only scratched the surface of what can be be done with the computation of best configurations in Section 5. The algorithms generalize easily to weights that are taken from an arbitrary ordered semiring (Golan, 1999; Borchardt and Vogler, 2003) and to computing minimal-weight rather than maximal-weight configurations. It is also useful in applications beyond semantic construction, e.g. in discourse parsing (Regneri et al., 2008).

**Acknowledgments.** We have benefited greatly from fruitful discussions on weighted tree grammars with Kevin Knight and Jonathan Graehl, and on discourse underspecification with Markus Egg. We also thank Christian Ebert, Marco Kuhlmann, Alex Lascarides, and the reviewers for their comments on the paper. Finally, we are deeply grateful to our former colleague Joachim Niehren, who was a great fan of tree automata before we even knew what they are.

## References


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

B. Borchardt and H. Vogler. 2003. Determinization of finite state weighted tree automata. *Journal of Automata, Languages and Combinatorics*, 8(3):417–463.

J. Bos. 1996. Predicate logic unplugged. In *Proceedings of the Tenth Amsterdam Colloquium*, pages 133–143.

R. P. Chaves. 2003. Non-redundant scope disambiguation in underspecified semantics. In *Proceedings of the 8th ESSLLI Student Session*, pages 47–58, Vienna.

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.

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*.

A. Copestake, D. Flickinger, C. Pollard, and I. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3:281–332.

C. Ebert. 2005. *Formal investigations of underspecified representations*. Ph.D. thesis, King's College, London.

M. Egg, A. Koller, and J. Niehren. 2001. The Constraint Language for Lambda Structures. *Logic, Language, and Information*, 10:457–485.

D. Flickinger, A. Koller, and S. Thater. 2005. A new well-formedness criterion for semantics debugging. In *Proceedings of the 12th HPSG Conference*, Lisbon.

J. S. Golan. 1999. *Semirings and their applications*. Kluwer, Dordrecht.

J. Graehl and K. Knight. 2004. Training tree transducers. In *HLT-NAACL 2004*, Boston.

D. Higgins and J. Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics*, 29(1).

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 J. Niehren. 2000. On underspecified processing of dynamic semantics. In *Proceedings of COLING-2000*, Saarbrücken.

A. Koller and O. Rambow. 2007. Relating dominance formalisms. In *Proceedings of the 12th Conference on Formal Grammar*, Dublin.

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*.

A. Koller and S. Thater. 2006. An improved redundancy elimination algorithm for underspecified descriptions. In *Proceedings of COLING/ACL-2006*, Sydney.

J. May and K. Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of HLT-NAACL*.

R. Nesson and S. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*.

J. Niehren and S. Thater. 2003. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *Proceedings of ACL 2003*.

S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1253–1257.

J. Pafel. 1997. Skopus und logische Struktur: Studien zum Quantorenskopus im Deutschen. Habilitationsschrift, Eberhard-Karls-Universität Tübingen.

M. Regneri, M. Egg, and A. Koller. 2008. Efficient processing of underspecified discourse representations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT) – Short Papers*, Columbus, Ohio.

U. Reyle. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10(1).

S. Shieber. 2006. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, Trento, Italy.

K. Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proceedings of the 16th conference on Computational linguistics*, pages 1175–1180, Morristown, NJ, USA. Association for Computational Linguistics.

M. Steedman. 2000. *The syntactic process*. MIT Press.

E. Vestre. 1991. An algorithm for generating non-redundant quantifier scopings. In *Proc. of EACL*, pages 251–256, Berlin.