Computing weakest readings

Alexander Koller
Cluster of Excellence
Saarland University
koller@mmci.uni-saarland.de

Stefan Thater
Dept. of Computational Linguistics
Saarland University
stth@coli.uni-saarland.de

Abstract

We present an efficient algorithm for computing the weakest readings of semantically ambiguous sentences. A corpus-based evaluation with a large-scale grammar shows that our algorithm reduces over 80% of sentences to one or two readings, in negligible runtime, and thus makes it possible to work with semantic representations derived by deep large-scale grammars.

1 Introduction

Over the past few years, there has been considerable progress in the ability of manually created large-scale grammars, such as the English Resource Grammar (ERG, Copestake and Flickinger (2000)) or the ParGram grammars (Butt et al., 2002), to parse wide-coverage text and assign it deep semantic representations. While applications should benefit from these very precise semantic representations, their usefulness is limited by the presence of semantic ambiguity: On the Rondane Treebank (Oepen et al., 2002), the ERG computes an average of several million semantic representations for each sentence, even when the syntactic analysis is fixed. The problem of appropriately selecting one of them to work with would ideally be solved by statistical methods (Higgins and Sadock, 2003) or knowledge-based inferences. However, no such approach has been worked out in sufficient detail to support the disambiguation of treebank sentences.

As an alternative, Bos (2008) proposes to compute the *weakest* reading of each sentence and then use it instead of the "true" reading of the sentence. This is based on the observation that the readings of a semantically ambiguous sentence are partially ordered with respect to logical entailment, and the weakest readings – the minimal (least informative) readings with respect to this order – only express "safe" information that is common to all other readings as well. However, when a sentence has millions of readings, finding the weakest reading is a hard problem. It is of course completely infeasible to compute all readings and compare all pairs for entailment; but even the best known algorithm in the literature (Gabsdil and Striegnitz, 1999) is only an optimization of this basic strategy, and would take months to compute the weakest readings for the sentences in the Rondane Treebank.

In this paper, we propose a new, efficient approach to the problem of computing weakest readings. We follow an *underspecification* approach to managing ambiguity: Rather than deriving all semantic representations from the syntactic analysis, we work with a single, compact underspecified semantic representation, from which the semantic representations can then be extracted by need. We then approximate entailment with a rewrite system that rewrites readings into logically weaker readings; the weakest readings are exactly those readings that cannot be rewritten into some other reading any more (the *relative normal forms*). We present an algorithm that computes the relative normal forms, and evaluate it on the underspecified descriptions that the ERG derives on a 624-sentence subcorpus of the Rondane Treebank. While the mean number of scope readings in the subcorpus is in the millions, our system computes on average 4.5 weakest readings for each sentence, in less than twenty milliseconds; over 80% of all sentences are reduced to at most two weakest readings. In other words, we make it feasible for the first time to build an application that uses the individual (weakest) semantic representations computed by the ERG, both in terms of the remaining ambiguity and in terms of performance. Our technique is not limited to the ERG, but should be applicable to other underspecification-based grammars as well.

Technically, we use underspecified descriptions that are regular tree grammars derived from dominance graphs (Althaus et al., 2003; Koller et al.,

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 30–39,
Uppsala, Sweden, 11-16 July 2010.

2008). We compute the weakest readings by intersecting these grammars with other grammars representing the rewrite rules. This approach can be used much more generally than just for the computation of weakest readings; we illustrate this by showing how a more general version of the redundancy elimination algorithm by Koller et al. (2008) can be seen as a special case of our construction. Thus our system can serve as a general framework for removing unintended readings from an underspecified representation.

The paper is structured as follows. Section 2 starts by reviewing related work. We recall dominance graphs, regular tree grammars, and the basic ideas of underspecification in Section 3, before we show how to compute weakest readings (Section 4) and logical equivalences (Section 5). In Section 6, we define a weakening rewrite system for the ERG and evaluate it on the Rondane Treebank. Section 7 concludes and points to future work.

2 Related work

The idea of deriving a single approximative semantic representation for ambiguous sentences goes back to Hobbs (1983); however, Hobbs only works his algorithm out for a restricted class of quantifiers, and his representations can be weaker than our weakest readings. Rules that weaken one reading into another were popular in the 1990s underspecification literature (Reyle, 1995; Monz and de Rijke, 2001; van Deemter, 1996) because they simplify logical reasoning with underspecified representations. From a linguistic perspective, Kempson and Cormack (1981) even go so far as to claim that the weakest reading should be taken as the "basic" reading of a sentence, and the other readings only seen as pragmatically licensed special cases.

The work presented here is related to other approaches that reduce the set of readings of an underspecified semantic representation (USR). Koller and Niehren (2000) showed how to strengthen a dominance constraint using information about anaphoric accessibility; later, Koller et al. (2008) presented and evaluated an algorithm for redundancy elimination, which removes readings from an USR based on logical equivalence. Our system generalizes the latter approach and applies it to a new inference problem (weakest readings) which they could not solve.

This paper builds closely upon Koller and Thater (2010), which lays the formal groundwork for the work presented here. Here we go beyond that paper by applying a concrete implementation of our RTG construction for weakest readings to a real-world grammar, evaluating the system on practical inputs, and combining weakest readings with redundancy elimination.

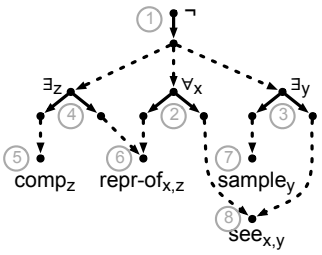

Figure 1: A dominance graph describing the five readings of the sentence "it is not the case that every representative of a company saw a sample."

3 Underspecification

This section briefly reviews two formalisms for specifying sets of trees: dominance graphs and regular tree grammars. Both of these formalisms can be used to model scope ambiguities compactly by regarding the semantic representations of a sentence as trees. Some example trees are shown in Fig. 2. These trees can be read as simplified formulas of predicate logic, or as formulas involving generalized quantifiers (Barwise and Cooper, 1981). Formally, we assume a ranked signature Σ of tree constructors $\{f, g, a, \ldots\}$, each of which is equipped with an arity $\mathsf{ar}(f) \geq 0$. We take a *(finite constructor) tree t* as a finite tree in which each node is labelled with a symbol of Σ, and the number of children of the node is exactly the arity of this symbol. For instance, the signature of the trees in Fig. 1 is $\{\forall_x|2, \exists_y|2, comp_z|0, \ldots\}$. Finite constructor trees can be seen as ground terms over Σ that respect the arities. We write $T(\Sigma)$ for the finite constructor trees over Σ.

3.1 Dominance graphs

A (labelled) dominance graph D (Althaus et al., 2003) is a directed graph that consists of a collection of trees called *fragments*, plus *dominance edges* relating nodes in different fragments. We distinguish the *roots* W_D of the fragments from their *holes*, which are the unlabelled leaves. We write $L_D : W_D \to \Sigma$ for the labeling function of D.

The basic idea behind using dominance graphs to model scope underspecification is to specify

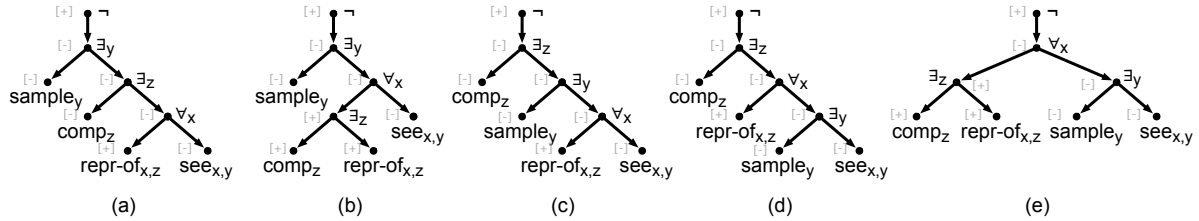

Figure 2: The five configurations of the dominance graph in Fig. 1.

the "semantic material" common to all readings as fragments, plus dominance relations between these fragments. An example dominance graph D is shown in Fig. 1. It represents the five readings of the sentence "it is not the case that every representative of a company saw a sample."

Each reading is encoded as a *(labeled) 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 (be an ancestor of) the target node in each configuration. The trees in Fig. 2 are the five labeled configurations of the example graph.

3.2 Regular tree grammars

Regular tree grammars (RTGs) are a general grammar formalism for describing languages of trees (Comon et al., 2007). An RTG is a 4-tuple $G = (S, N, \Sigma, P)$, where N and Σ are nonterminal and terminal alphabets, $S \in N$ is the start symbol, and P is a finite set of production rules. Unlike in context-free string grammars (which look superficially the same), the terminal symbols are tree constructors from Σ. The production rules are of the form $A \to t$, where A is a nonterminal and t is a tree from $T(\Sigma \cup N)$; nonterminals count as having arity zero, i.e. they must label leaves. A derivation starts with a tree containing a single node labeled with S. Then in each step of the derivation, some leaf u which is labelled with a nonterminal A is expanded with a rule $A \to t$; this results in a new tree in which u has been replaced by t, and the derivation proceeds with this new tree. The *language* $L(G)$ generated by the grammar is the set of all trees in $T(\Sigma)$ that can be derived in this way.

Fig. 3 shows an RTG as an example. This grammar uses sets of root names from D as nonterminal symbols, and generates exactly the five configurations of the graph in Fig. 1.

The languages that can be accepted by regular tree grammars are called *regular tree languages*

$$
\begin{aligned}
\{1,2,3,4,5,6,7,8\} &\to \neg(\{2,3,4,5,6,7,8\}) \\
\{2,3,4,5,6,7,8\} &\to \forall_x(\{4,5,6\},\{3,7,8\}) \\
\{2,3,4,5,6,7,8\} &\to \exists_y(\{7\},\{2,4,5,6,8\}) \\
\{2,3,4,5,6,7,8\} &\to \exists_z(\{5\},\{2,3,6,7,8\}) \\
\{2,4,5,6,8\} &\to \forall_x(\{4,5,6\},\{8\}) \\
&\quad \mid \exists_z(\{5\},\{2,6,8\}) \\
\{2,3,6,7,8\} &\to \forall_x(\{6\},\{3,7,8\}) \\
&\quad \mid \exists_y(\{7\},\{2,6,8\}) \\
\{2,6,8\} &\to \forall_x(\{6\},\{8\}) \\
\{3,7,8\} &\to \exists_y(\{7\},\{8\}) \\
\{4,5,6\} &\to \exists_z(\{5\},\{6\}) \\
\{5\} \to \mathit{comp}_z \qquad & \{7\} \to \mathit{sample}_y \\
\{6\} \to \mathit{repr\text{-}of}_{x,z} \qquad & \{8\} \to \mathit{see}_{x,y}
\end{aligned}
$$

Figure 3: A regular tree grammar that generates the five trees in Fig. 2.

(RTLs), and regular tree grammars are equivalent to *finite tree automata*, which are defined essentially like the well-known finite string automata, except that they assign states to the nodes in a tree rather than the positions in a string. Regular tree languages enjoy many of the closure properties of regular string languages. In particular, we will later exploit that RTLs are closed under intersection and complement.

3.3 Dominance graphs as RTGs

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

Every hypernormally connected dominance graph D can be automatically translated into an equivalent RTG G_D that generates exactly the same configurations (Koller et al., 2008); the RTG in Fig. 3 is an example. The nonterminals of G_D are

always hnc subgraphs of D. In the worst case, G_D can be exponentially bigger than D, but in practice it turns out that the grammar size remains manageable: even the RTG for the most ambiguous sentence in the Rondane Treebank, which has about 4.5×10^{12} scope readings, has only about 75 000 rules and can be computed in a few seconds.

4 Computing weakest readings

Now we are ready to talk about computing the weakest readings of a hypernormally connected dominance graph. We will first explain how we approximate logical weakening with rewrite systems. We will then discuss how weakest readings can be computed efficiently as the *relative normal forms* of these rewrite systems.

4.1 Weakening rewrite systems

The different readings of a sentence with a scope ambiguity are not a random collection of formulas; they are partially ordered with respect to logical entailment, and are structurally related in a way that allows us to model this entailment relation with simpler technical means.

To illustrate this, consider the five configurations in Fig. 2. The formula represented by (d) logically entails (c); we say that (c) is a *weaker* reading than (d) because it is satisfied by more models. Similar entailment relations hold between (d) and (e), (e) and (b), and so on (see also Fig. 5). We can define the *weakest readings* of the dominance graph as the minimal elements of the entailment order; in the example, these are (b) and (c). Weakest readings capture "safe" information in that whichever reading of the sentence the speaker had in mind, any model of this reading also satisfies at least one weakest reading; in the absence of convincing disambiguation methods, they can therefore serve as a practical approximation of the intended meaning of the sentence.

A naive algorithm for computing weakest readings would explicitly compute the entailment order, by running a theorem prover on each pair of configurations, and then pick out the minimal elements. But this algorithm is quadratic in the number of configurations, and therefore impractically slow for real-life sentences.

Here we develop a fast algorithm for this problem. The fundamental insight we exploit is that entailment among the configurations of a dominance graph can be approximated with rewriting rules (Baader and Nipkow, 1999). Consider the relation between (d) and (c). We can explain that (d) entails (c) by observing that (c) can be built from (d) by exchanging the positions of the adjacent quantifiers $\forall_x$ and $\exists_y$; more precisely, by applying the following rewrite rule:

$$[-]\ \forall_x(Q, \exists_y(P,R)) \to \exists_y(P, \forall_x(Q,R)) \qquad (1)$$

The body of the rule specifies that an occurrence of $\forall_x$ which is the direct parent of an occurrence of $\exists_y$ may change positions with it; the subformulas P, Q, and R must be copied appropriately. The annotation $[-]$ specifies that we must only apply the rule to subformulas in negative logical polarity: If the quantifiers in (d) were not in the scope of a negation, then applying the rule would actually make the formula *stronger*. We say that the rule (1) is logically *sound* because applying it to a subformula with the correct polarity of some configuration t always makes the result t' logically weaker than t.

We formalize these rewrite systems as follows. We assume a finite *annotation alphabet* Ann with a special *starting annotation* $\mathsf{a}_0 \in \mathsf{Ann}$; in the example, we had $\mathsf{Ann} = \{+,-\}$ and $\mathsf{a}_0 = +$. We also assume an *annotator function* $\mathsf{ann} : \mathsf{Ann} \times \Sigma \times \mathbb{N} \to \mathsf{Ann}$. The function ann can be used to traverse a tree top-down and compute the annotation of each node from the annotation of its parent: Its first argument is the annotation and its second argument the node label of the parent, and the third argument is the position of the child among the parent's children. In our example, the annotator ann models logical polarity by mapping, for instance, $\mathsf{ann}(+,\exists_z,1) = \mathsf{ann}(+,\exists_z,2) = \mathsf{ann}(+,\exists_y,2) = +$, $\mathsf{ann}(-,\exists_z,1) = \mathsf{ann}(-,\exists_z,2) = \mathsf{ann}(+,\forall_x,1) = -$, etc. We have labelled each node of the configurations in Fig. 1 with the annotations that are computed in this way.

Now we can define an *annotated rewrite system* R to be a finite set of pairs (a,r) where a is an annotation and r is an ordinary rewrite rule. The rule (1) above is an example of an annotated rewrite rule with $a = -$. A rewrite rule (a,r) can be applied at the node u of a tree t if ann assigns the annotation a to u and r is applicable at u as usual. The rule then rewrites t as described above. In other words, annotated rewrite systems are rewrite systems where rule applications are restricted to subtrees with specific annotations. We write $t \to_R t'$ if some rule of R can be applied at a node of t, and the result of rewriting is t'. The rewrite system R is called *linear*

if every variable that occurs on the left-hand side of a rule occurs on its right-hand side exactly once.

4.2 Relative normal forms

The rewrite steps of a sound weakening rewrite system are related to the entailment order: Because every rewrite step transforms a reading into a weaker reading, an actual weakest readings must be such that there is no other configuration into which it can be rewritten. The converse is not always true, i.e. there can be non-rewritable configurations that are not weakest readings, but we will see in Section 6 that this approximation is good enough for practical use. So one way to solve the problem of computing weakest readings is to find readings that cannot be rewritten further.

One class of configurations that "cannot be rewritten" with a rewrite system R is the set of *normal forms* of R, i.e. those configurations to which no rule in R can be applied. In our example, (b) and (c) are indeed normal forms with respect to a rewrite system that consists only of the rule (1). However, this is not exactly what we need here. Consider a rewrite system that also contains the following annotated rewrite rule, which is also sound for logical entailment:

$$[+]\ \neg(\exists_z(P,Q)) \to \exists_z(P,\neg(Q)), \qquad (2)$$

This rule would allow us to rewrite the configuration (c) into the tree $\exists_z(\mathsf{comp}_z, \neg(\exists_y(\mathsf{sample}_y, \forall_x(\mathsf{repr{-}of}_{x,z}, \mathsf{see}_{x,y}))))$. But this is no longer a configuration of the graph. If we were to equate weakest readings with normal forms, we would erroneously classify (c) as not being a weakest reading. The correct concept for characterizing weakest readings in terms of rewriting is that of a *relative normal form*. We define a configuration t of a dominance graph D to be a R-relative normal form of (the configurations of) D iff there is no other configuration t' of D such that $t \to_R t'$. These are the configurations that can't be weakened further without obtaining a tree that is no longer a configuration of D. In other words, if R approximates entailment, then the R-relative normal forms approximate the weakest readings.

4.3 Computing relative normal forms

We now show how the relative normal forms of a dominance graph can be computed efficiently. For lack of space, we only sketch the construction and omit all proofs. Details can be found in Koller and Thater (2010).

The key idea of the construction is to represent the relation $\to_R$ in terms of a *context tree transducer* M, and characterize the relative normal forms of a tree language L in terms of the pre-image of L under M. Like ordinary regular tree transducers (Comon et al., 2007), context tree transducers read an input tree, assigning states to the nodes, while emitting an output tree. But while ordinary transducers read the input tree symbol by symbol, a context tree transducer can read multiple symbols at once. In this way, they are equivalent to the extended left-hand side transducers of Graehl et al. (2008).

We will now define context tree transducers. Let Σ be a ranked signature, and let X_m be a set of m variables. We write $\mathrm{Con}^{(m)}(\Sigma)$ for the *contexts* with m holes, i.e. those trees in $T(\Sigma \cup X_m)$ in which each element of X_m occurs exactly once, and always as a leaf. If $C \in \mathrm{Con}^{(m)}(\Sigma)$, then $C[t_1,\ldots,t_m] = C[t_1/x_1,\ldots,t_m/x_m]$, where $x_1,\ldots,x_m$ are the variables from left to right.

A *(top-down) context tree transducer* from Σ to Δ is a 5-tuple $M = (Q,\Sigma,\Delta,q_0,\delta)$. Σ and Δ are ranked signatures, Q is a finite set of states, and $q_0 \in Q$ is the start state. δ is a finite set of transition rules of the form $q(C[x_1,\ldots,x_n]) \to D[q_1(x_{i_1}),\ldots,q_m(x_{i_m})]$, where $C \in \mathrm{Con}^{(n)}(\Sigma)$ and $D \in \mathrm{Con}^{(m)}(\Delta)$.

If $t \in T(\Sigma \cup \Delta \cup Q)$, then we say that M derives t' in one step from t, $t \to_M t'$, if t is of the form $C'[q(C[t_1,\ldots,t_n])]$ for some $C' \in \mathrm{Con}^{(1)}(\Sigma)$, t' is of the form $C'[D[q_1(t_{i_1}),\ldots,q_m(t_{i_m})]]$, and there is a rule $q(C[x_1,\ldots,x_n]) \to D[q_1(x_{i_1}),\ldots,q_m(x_{i_m})]$ in δ. The *derivation relation* $\to_M^*$ is the reflexive, transitive closure of $\to_M$. The *translation relation* τ_M of M is

$$\tau_M = \{(t,t') \mid t \in T(\Sigma) \text{ and } t' \in T(\Delta) \text{ and } q_0(t) \to^* t'\}.$$

For each linear annotated rewrite system R, we can now build a context tree transducer M_R such that $t \to_R t'$ iff $(t,t') \in \tau_{M_R}$. The idea is that M_R traverses t from the root to the leaves, keeping track of the current annotation in its state. M_R can nondeterministically choose to either copy the current symbol to the output tree unchanged, or to apply a rewrite rule from R. The rules are built in such a way that in each run, exactly one rewrite rule must be applied.

We achieve this as follows. M_R takes as its states the set $\{\bar{q}\} \cup \{q^a \mid a \in \mathsf{Ann}\}$ and as its start state the state q^{a_0}. If M_R reads a node u in state q^a, this means that the annotator assigns annotation a to u and M_R will rewrite a subtree at or

below u. If M_R reads u in state $\bar{q}$, this means that M_R will copy the subtree below u unchanged because the rewriting has taken place elsewhere. Thus M_R has three types of rewrite rules. First, for any $f \in \Sigma$, we have a rule $\bar{q}(f(x_1,\ldots,x_n)) \to f(\bar{q}(x_1),\ldots,\bar{q}(x_n))$. Second, for any f and $1 \leq i \leq n$, we have a rule $q^a(f(x_1,\ldots,x_n)) \to f(\bar{q}(x_1),\ldots,q^{\mathsf{ann}(a,f,i)}(x_i),\ldots,\bar{q}(x_n))$, which nondeterministically chooses under which child the rewriting should take place, and assigns it the correct annotation. Finally, we have a rule $q^a(C[x_1,\ldots,x_n]) \to C'[\bar{q}(x_{i_1}),\ldots,\bar{q}(x_{i_n})]$ for every rewrite rule $C[x_1,\ldots,x_n] \to C'[x_{i_1},\ldots,x_{i_n}]$ with annotation a in R.

Now let's put the different parts together. We know that for each hnc dominance graph D, there is a regular tree grammar G_D such that $L(G_D)$ is the set of configurations of D. Furthermore, the pre-image $\tau_M^{-1}(L) = \{t \mid \text{exists } t' \in L \text{ with } (t,t') \in \tau_M\}$ of a regular tree language L is also regular (Koller and Thater, 2010) if M is linear, and regular tree languages are closed under intersection and complement (Comon et al., 2007). So we can compute another RTG G' such that

$$L(G') = L(G_D) \cap \overline{\tau_{M_R}^{-1}(L(G_D))}.$$

$L(G')$ consists of the members of $L(G_D)$ which cannot be rewritten by M_R into members of $L(G_D)$; that is, $L(G')$ is exactly the set of R-relative normal forms of D. In general, the complement construction requires exponential time in the size of M_R and G_D. However, it can be shown that if the rules in R have at most depth two and G_D is deterministic, then the entire above construction can be computed in time $O(|G_D| \cdot |R|)$ (Koller and Thater, 2010).

In other words, we have shown how to compute the weakest readings of a hypernormally connected dominance graph D, as approximated by a weakening rewrite system R, in time linear in the size of G_D and linear in the size of R. This is a dramatic improvement over the best previous algorithm, which was quadratic in $|\mathsf{conf}(D)|$.

4.4 An example

Consider an annotated rewrite system that contains rule (1) plus the following rewrite rule:

$$[-]\ \exists_z(P, \forall_x(Q,R)) \to \forall_x(\exists_z(P,Q),R) \qquad (3)$$

This rewrite system translates into a top-down context tree transducer M_R with the following transition rules, omitting most rules of the first two types for lack of space.

$$\begin{aligned}
q^-(\forall_x(x_1,\exists_y(x_2,x_3))) &\to \exists_y(\bar{q}(x_2),\forall_x(\bar{q}(x_1),\bar{q}(x_3)))\\
q^-(\exists_y(x_1,\forall_x(x_2,x_3))) &\to \forall_x(\exists_y(\bar{q}(x_1),\bar{q}(x_2)),\bar{q}(x_3))\\
\bar{q}(\neg(x_1)) &\to \neg(\bar{q}(x_1))\\
q^+(\neg(x_1)) &\to \neg(q^-(x_1))\\
\bar{q}(\forall_x(x_1,x_2)) &\to \forall_x(\bar{q}(x_1),\bar{q}(x_2))\\
q^+(\forall_x(x_1,x_2)) &\to \forall_x(\bar{q}(x_1),q^+(x_2))\\
q^+(\forall_x(x_1,x_2)) &\to \forall_x(q^-(x_1),\bar{q}(x_2)) \quad \ldots
\end{aligned}$$

$$\begin{aligned}
\{1,2,3,4,5,6,7,8\}_F &\to \neg(\{2,3,4,5,6,7,8\}_F)\\
\{2,3,4,5,6,7,8\}_F &\to \exists_y(\{7\}_{\{\bar{q}\}},\{2,4,5,6,8\}_F)\\
&\quad\ \mid \exists_z(\{5\}_{\{\bar{q}\}},\{2,3,6,7,8\}_F)\\
\{2,3,6,7,8\}_F &\to \exists_y(\{7\}_{\{\bar{q}\}},\forall_x(\{6\}_{\{\bar{q}\}},\{8\}_{\{\bar{q}\}}))\\
\{2,4,5,6,8\}_F &\to \forall_x(\{4,5,6\}_{\{\bar{q}\}},\{8\}_{\{\bar{q}\}})\\
\{4,5,6\}_{\{\bar{q}\}} &\to \exists_z(\{5\}_{\{\bar{q}\}},\{6\}_{\{\bar{q}\}})\\
\{5\}_{\{\bar{q}\}} &\to \mathit{comp}_z \qquad \{6\}_{\{\bar{q}\}} \to \mathit{repr\text{-}of}_{x,z}\\
\{7\}_{\{\bar{q}\}} &\to \mathit{sample}_y \quad \{8\}_{\{\bar{q}\}} \to \mathit{see}_{x,y}
\end{aligned}$$

Figure 4: RTG for the weakest readings of Fig. 1.

The grammar G' for the relative normal forms is shown in Fig. 4 (omitting rules that involve unproductive nonterminals). We obtain it by starting with the example grammar G_D in Fig. 3; then computing a deterministic RTG G_R for $\tau_{M_R}^{-1}(L(G_D))$; and then intersecting the complement of G_R with G_D. The nonterminals of G' are subgraphs of D, marked either with a set of states of M_R or the symbol F, indicating that G_R had no production rule for a given left-hand side. The start symbol of G' is marked with F because G' should only generate trees that G_R can*not* generate. As expected, G' generates precisely two trees, namely (b) and (c).

5 Redundancy elimination, revisited

The construction we just carried out – characterize the configurations we find interesting as the relative normal forms of an annotated rewrite system R, translate it into a transducer M_R, and intersect $\mathsf{conf}(D)$ with the complement of the pre-image under M_R – is more generally useful than just for the computation of weakest readings. We illustrate this on the problem of *redundancy elimination* (Vestre, 1991; Chaves, 2003; Koller et al., 2008) by showing how a variant of the algorithm of Koller et al. (2008) falls out of our technique as a special case.

Redundancy elimination is the problem of computing, from a dominance graph D, another dominance graph D' such that $\mathsf{conf}(D') \subseteq \mathsf{conf}(D)$ and

every formula in $\mathsf{conf}(D)$ is *logically equivalent* to some formula in $\mathsf{conf}(D')$. We can approximate logical equivalence using a finite system of equations such as

$$\exists_y(P, \exists_z(Q,R)) = \exists_z(Q, \exists_y(P,R)), \qquad (4)$$

indicating that $\exists_y$ and $\exists_z$ can be permuted without changing the models of the formula.

Following the approach of Section 4, we can solve the redundancy elimination problem by transforming the equation system into a rewrite system R such that $t \to_R t'$ implies that t and t' are equivalent. To this end, we assume an arbitrary linear order $<$ on Σ, and orient all equations into rewrite rules that respect this order. If we assume $\exists_y < \exists_z$, the example rule (4) translates into the annotated rewrite rules

$$[a]\ \exists_z(P, \exists_y(Q,R)) \to \exists_y(Q, \exists_z(P,R)) \qquad (5)$$

for all annotations $a \in \mathsf{Ann}$; logical equivalence is not sensitive to the annotation. Finally, we can compute the relative normal forms of $\mathsf{conf}(D)$ under this rewrite system as above. The result will be an RTG G' describing a subset of $\mathsf{conf}(D)$. Every tree t in $\mathsf{conf}(D)$ that is not in $L(G')$ is equivalent to some tree t' in $L(G')$, because if t could not be rewritten into such a t', then t would be in relative normal form. That is, the algorithm solves the redundancy elimination problem. Furthermore, if the oriented rewrite system is confluent (Baader and Nipkow, 1999), no two trees in $L(G')$ will be equivalent to each other, i.e. we achieve complete reduction in the sense of Koller et al. (2008).

This solution shares much with that of Koller et al. (2008), in that we perform redundancy elimination by intersecting tree grammars. However, the construction we present here is much more general: The algorithmic foundation for redundancy elimination is now exactly the same as that for weakest readings, we only have to use an equivalence-preserving rewrite system instead of a weakening one. This new formal clarity also simplifies the specification of certain equations, as we will see in Section 6.

In addition, we can now combine the weakening rules (1), (3), and (5) into a single rewrite system, and then construct a tree grammar for the relative normal forms of the combined system. This algorithm performs redundancy elimination and computes weakest readings at the same time, and in our example retains only a single configuration, namely

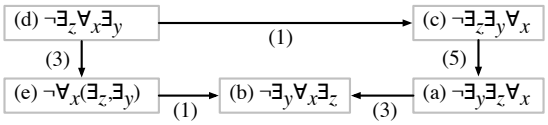

Figure 5: Structure of the configuration set of Fig. 1 in terms of rewriting.

(b); the configuration (c) is rejected because it can be rewritten to (a) with (5). The graph in Fig. 5 illustrates how the equivalence and weakening rules conspire to exclude all other configurations.

6 Evaluation

In this section, we evaluate the effectiveness and efficiency of our weakest readings algorithm on a treebank. We compute RTGs for all sentences in the treebank and measure how many weakest readings remain after the intersection, and how much time this computation takes.

Resources. For our experiment, we use the Rondane treebank (version of January 2006), a "Redwoods style" (Oepen et al., 2002) treebank containing underspecified representations (USRs) in the MRS formalism (Copestake et al., 2005) for sentences from the tourism domain.

Our implementation of the relative normal forms algorithm is based on Utool (Koller and Thater, 2005), which (among other things) can translate a large class of MRS descriptions into hypernormally connected dominance graphs and further into RTGs as in Section 3. The implementation exploits certain properties of RTGs computed from dominance graphs to maximize efficiency. We will make this implementation publically available as part of the next Utool release.

We use Utool to automatically translate the 999 MRS descriptions for which this is possible into RTGs. To simplify the specification of the rewrite systems, we restrict ourselves to the subcorpus in which all scope-taking operators (labels with arity > 0) occur at least ten times. This subset contains 624 dominance graphs. We refer to this subset as "RON10."

Signature and annotations. For each dominance graph D that we obtain by converting an MRS description, we take G_D as a grammar over the signature $\Sigma = \{f_u \mid u \in W_D, f = L_D(u)\}$. That is, we distinguish possible different occurrences of the same symbol in D by marking each occur-

rence with the name of the node. This makes G_D a deterministic grammar.

We then specify an annotator over Σ that assigns polarities for the weakening rewrite system. We distinguish three polarities: $+$ for positive occurrences, $-$ for negative occurrences (as in predicate logic), and $\perp$ for contexts in which a weakening rule neither weakens or strengthens the entire formula. The starting annotation is $+$.

Finally, we need to decide upon each scope-taking operator's effects on these annotations. To this end, we build upon Barwise and Cooper's (1981) classification of the monotonicity properties of determiners. A determiner is *upward (downward) monotonic* if making the denotation of the determiner's argument bigger (smaller) makes the sentence logically weaker. For instance, *every* is downward monotonic in its first argument and upward monotonic in its second argument, i.e. *every girl kissed a boy* entails *every blond girl kissed someone*. Thus $\mathsf{ann}(\mathsf{every}_u, a, 1) = -a$ and $\mathsf{ann}(\mathsf{every}_u, a, 2) = a$ (where u is a node name as above). There are also determiners with non-monotonic argument positions, which assign the annotation $\perp$ to this argument. Negation reverses positive and negative polarity, and all other non-quantifiers simply pass on their annotation to the arguments.

Weakest readings. We use the following weakening rewrite system for our experiment, where $i \in \{1,2\}$:

1. [+] $(\mathsf{E}/i, \mathsf{D}/1)$, $(\mathsf{D}/2, \mathsf{D}/1)$
2. [+] $(\mathsf{E}/i, \mathsf{P}/1)$, $(\mathsf{D}/2, \mathsf{P}/1)$
3. [+] $(\mathsf{E}/i, \mathsf{A}/2)$, $(\mathsf{D}/1, \mathsf{A}/2)$
4. [+] $(\mathsf{A}/2, \mathsf{N}/1)$
5. [+] $(\mathsf{N}/1, \mathsf{E}/i)$, $(\mathsf{N}/1, \mathsf{D}/2)$
6. [+] $(\mathsf{E}/i, \mathsf{M}/1)$, $(\mathsf{D}/1, \mathsf{M}/1)$

Here the symbols E, D, etc. stand for classes of labels in Σ, and a rule schema $[a]$ $(\mathsf{C}/i, \mathsf{C}'/k)$ is to be read as shorthand for a set of rewrite rules which rearrange a tree where the i-th child of a symbol from C is a symbol from C' into a tree where the symbol from C becomes the k-th child of the symbol from C'. For example, because we have $\mathsf{all}_u \in \mathsf{A}$ and $\mathsf{not}_v \in \mathsf{N}$, Schema 4 licenses the following annotated rewrite rule:

$$[+]\ \mathsf{all}_u(P, \mathsf{not}_v(Q)) \rightarrow \mathsf{not}_v(\mathsf{all}_u(P, Q)).$$

We write E and D for existential and definite determiners. P stands for proper names and pronouns, A stands for universal determiners like *all* and *each*, N for the negation *not,* and M for modal operators like *can* or *would*. M also includes intensional verbs like *have to* and *want*. Notice that while the reverse rules are applicable in negative polarities, no rules are applicable in polarity $\perp$.

Rule schema 1 states, for instance, that the specific (wide-scope) reading of the indefinite in *the president of a company* is logically stronger than the reading in which *a company* is within the restriction of the definite determiner. The schema is intuitively plausible, and it can also be proved to be logically sound if we make the standard assumption that the definite determiner *the* means "exactly one" (Montague, 1974). A similar argument applies to rule schema 2.

Rule schema 3 encodes the classical entailment (1). Schema 4 is similar to the rule (2). Notice that it is not, strictly speaking, logically sound; however, because strong determiners like *all* or *every* carry a presupposition that their restrictions have a non-empty denotation (Lasersohn, 1993), the schema becomes sound for all instances that can be expressed in natural language. Similar arguments apply to rule schemas 5 and 6, which are potentially unsound for subtle reasons involving the logical interpretation of intensional expressions. However, these cases of unsoundness did not occur in our test corpus.

Redundancy elimination. In addition, we assume the following equation system for redundancy elimination for $i, j \in \{1,2\}$ and $k \in \mathbb{N}$ (again written in an analogous shorthand as above):

7. $\mathsf{E}/i = \mathsf{E}/j$
8. $\mathsf{D}/1 = \mathsf{E}/i,\ \mathsf{E}/i = \mathsf{D}/1$
9. $\mathsf{D}/1 = \mathsf{D}/1$
10. $\Sigma/k = \mathsf{P}/2$

These rule schemata state that permuting existential determiners with each other is an equivalence transformation, and so is permuting definite determiners with existential and definite determiners if one determiner is the second argument (in the scope) of a definite. Schema 10 states that proper names and pronouns, which the ERG analyzes as scope-bearing operators, can permute with any other label.

We orient these equalities into rewrite rules by ordering symbols in P before symbols that are not

	All	KRT08	RE	RE+WR
#conf = 1	8.5%	23.4%	34.9%	66.7%
#conf ≤ 2	20.5%	40.9%	57.9%	80.6%
avg(#conf)	3.2M	7603.1	119.0	4.5
med(#conf)	25	4	2	1
runtime	8.1s	9.4s	8.7s	9.1s

Figure 6: Analysis of the numbers of configurations in RON10.

in P, and otherwise ordering a symbol f_u before a symbol g_v if $u < v$ by comparison of the (arbitrary) node names.

Results. We used these rewrite systems to compute, for each USR in RON10, the number of all configurations, the number of configurations that remain after redundancy elimination, and the number of weakest readings (i.e., the relative normal forms of the combined equivalence and weakening rewrite systems). The results are summarized in Fig. 6. By computing weakest readings (WR), we reduce the ambiguity of over 80% of all sentences to one or two readings; this is a clear improvement even over the results of the redundancy elimination (RE). Computing weakest readings reduces the mean number of readings from several million to 4.5, and improves over the RE results by a factor of 30. Notice that the RE algorithm from Section 5 is itself an improvement over Koller et al.'s (2008) system ("KRT08" in the table), which could not process the rule schema 10.

Finally, computing the weakest readings takes only a tiny amount of extra runtime compared to the RE elimination or even the computation of the RTGs (reported as the runtime for "All").[1] This remains true on the entire Rondane corpus (although the reduction factor is lower because we have no rules for the rare scope-bearers): RE+WR computation takes 32 seconds, compared to 30 seconds for RE. In other words, our algorithm brings the semantic ambiguity in the Rondane Treebank down to practically useful levels at a mean runtime investment of a few milliseconds per sentence.

It is interesting to note how the different rule schemas contribute to this reduction. While the instances of Schemata 1 and 2 are applicable in 340 sentences, the other schemas 3–6 together are only applicable in 44 sentences. Nevertheless, where these rules do apply, they have a noticeable effect: Without them, the mean number of configurations in RON10 after RE+WR increases to 12.5.

[1] Runtimes were measured on an Intel Core 2 Duo CPU at 2.8 GHz, under MacOS X 10.5.6 and Apple Java 1.5.0_16, after allowing the JVM to just-in-time compile the bytecode.

7 Conclusion

In this paper, we have shown how to compute the weakest readings of a dominance graph, characterized by an annotated rewrite system. Evaluating our algorithm on a subcorpus of the Rondane Treebank, we reduced the mean number of configurations of a sentence from several million to 4.5, in negligible runtime. Our algorithm can be applied to other problems in which an underspecified representation is to be disambiguated, as long as the remaining readings can be characterized as the relative normal forms of a linear annotated rewrite system. We illustrated this for the case of redundancy elimination.

The algorithm presented here makes it possible, for the first time, to derive a single meaningful semantic representation from the syntactic analysis of a deep grammar on a large scale. In the future, it will be interesting to explore how these semantic representations can be used in applications. For instance, it seems straightforward to adapt MacCartney and Manning's (2008) "natural logic"-based Textual Entailment system, because our annotator already computes the polarities needed for their monotonicity inferences. We could then perform such inferences on (cleaner) semantic representations, rather than strings (as they do).

On the other hand, it may be possible to reduce the set of readings even further. We retain more readings than necessary in many treebank sentences because the combined weakening and equivalence rewrite system is not confluent, and therefore may not recognize a logical relation between two configurations. The rewrite system could be made more powerful by running the Knuth-Bendix completion algorithm (Knuth and Bendix, 1970). Exploring the practical tradeoff between the further reduction in the number of remaining configurations and the increase in complexity of the rewrite system and the RTG would be worthwhile.

Acknowledgments. We are indebted to Joachim Niehren, who pointed out a crucial simplification in the algorithm to us. We also thank our reviewers for their constructive comments.

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.

F. Baader and T. Nipkow. 1999. *Term rewriting and all that*. Cambridge University Press.

J. Barwise and R. Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.

J. Bos. 2008. Let's not argue about semantics. In *Proceedings of the 6th international conference on Language Resources and Evaluation (LREC 2008)*.

M. Butt, H. Dyvik, T. Holloway King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*.

R. P. Chaves. 2003. Non-redundant scope disambiguation in underspecified semantics. In *Proceedings of the 8th ESSLLI Student Session*.

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 *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC)*.

A. Copestake, D. Flickinger, C. Pollard, and I. Sag. 2005. Minimal recursion semantics: An introduction. *Journal of Language and Computation*.

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

M. Gabsdil and K. Striegnitz. 1999. Classifying scope ambiguities. In *Proceedings of the First Intl. Workshop on Inference in Computational Semantics*.

J. Graehl, K. Knight, and J. May. 2008. Training tree transducers. *Computational Linguistics*, 34(3):391–427.

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

J. Hobbs. 1983. An improper treatment of quantification in ordinary English. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL'83)*.

R. Kempson and A. Cormack. 1981. Ambiguity and quantification. *Linguistics and Philosophy*, 4:259–309.

D. Knuth and P. Bendix. 1970. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford.

A. Koller and J. Niehren. 2000. On underspecified processing of dynamic semantics. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*.

A. Koller and S. Thater. 2005. Efficient solving and exploration of scope ambiguities. In *ACL-05 Demonstration Notes*, Ann Arbor.

A. Koller and S. Thater. 2010. Computing relative normal forms in regular tree languages. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA)*.

A. Koller, J. Niehren, and S. Thater. 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *Proceedings of the 10th EACL*.

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

P. Lasersohn. 1993. Existence presuppositions and background knowledge. *Journal of Semantics*, 10:113–122.

B. MacCartney and C. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*.

R. Montague. 1974. The proper treatment of quantification in ordinary English. In R. Thomason, editor, *Formal Philosophy. Selected Papers of Richard Montague*. Yale University Press, New Haven.

C. Monz and M. de Rijke. 2001. Deductions with meaning. In Michael Moortgat, editor, *Logical Aspects of Computational Linguistics, Third International Conference (LACL'98)*, volume 2014 of *LNAI*. Springer-Verlag, Berlin/Heidelberg.

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)*.

Uwe Reyle. 1995. On reasoning with ambiguities. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*.

K. van Deemter. 1996. Towards a logic of ambiguous expressions. In *Semantic Ambiguity and Underspecification*. CSLI Publications, Stanford.

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