Computing Logical Form on Regulatory Texts[*]

Nikhil Dinesh
Artificial Intelligence Center
SRI International
Menlo Park, CA - 94025
dinesh@ai.sri.com

Aravind Joshi and **Insup Lee**
Department of Computer Science
University of Pennsylvania
Philadelphia, PA - 19104
{joshi,lee}@seas.upenn.edu

Abstract

The computation of logical form has been proposed as an intermediate step in the translation of sentences to logic. Logical form encodes the resolution of scope ambiguities. In this paper, we describe experiments on a modest-sized corpus of regulation annotated with a novel variant of logical form, called *abstract syntax trees* (ASTs). The main step in computing ASTs is to order scope-taking operators. A learning model for ranking is adapted for this ordering. We design features by studying the problem of comparing the scope of one operator to another. The scope comparisons are used to compute ASTs, with an F-score of 90.6% on the set of ordering decisons.

1 Introduction

May (1985) argued for a level of *logical form* as a prelude to translating sentences to logic. Just as a parse tree determines the constituent structure of a sentence, a logical form of a sentence represents one way of resolving scope ambiguities. The level of logical form is an appealing layer of modularity; it allows us to take a step beyond parsing in studying scope phenomenon, and yet, avoid the open problem of fully translating sentences to logic.

Data-driven analyses of scope have been of interest in psycholinguistics (Kurtzman and MacDonald, 1993) and more recently in NLP (Srinivasan and Yates, 2009). The focus has typically been on predicting the preferred scopal ordering of sentences with two quantifying determiners, for example, in the sentence “every kid climbed a tree”. In the related problem of translating database queries to logic, Zettlemoyer and Collins (2009) and Wong and Mooney (2007) consider the scope of adjectives in addition to determiners, for example the scope of “cheapest” in the noun phrase “the cheapest flights from Boston to New York”. To our knowledge, empirical studies of scope have been restricted to phenomenon between and within noun phrases.

In this paper, we describe experiments on a novel annotation of scope phenomenon in regulatory texts – Section 610 of the Food and Drug Administration’s Code of Federal Regulations[1] (FDA CFR). Determiners, modals, negation, and verb phrase modifiers are the main scope-taking operators. We have annotated 195 sentences with a variant of logical form, called *abstract syntax trees* (ASTs). Our focus is on the problem of computing the AST, given a (variant of a) parse tree of a sentence.

The long term goal of this work is to assist in the translation of regulation to logic, for the application of conformance checking. The problem is to formally determine whether an organization conforms to regulation, by checking the organization’s records using the logical translation of regulation. Conformance checking has been of interest in a variety of regulatory contexts, and examples include privacy policy (Barth et al., 2006; Jones and Sergot, 1992; Anderson, 1996) and business contracts (Governatori et al., 2006; Grosof et al., 1999).

We now discuss some problems that arise in defin-

[*]This research was supported in part by ONR MURI N00014-07-1-0907, NSF CNS-1035715, NSF IIS 07-05671, and SRI International.

[1]http://www.gpoaccess.gov/cfr/index.html

ing logical form and the assumptions that we make to circumvent these problems.

1.1 Problems and Assumptions

A key assumption of logical form is that the translation from language to logic is syntax-based. As a result, the logic needs to be expressive enough to accomodate a syntactic translation. There is no consensus logic for constructs, such as, plurals, purpose clauses, and certain modals. This leads to the following problem in defining logical form.
How do we define the logical form of a sentence, without defining the logic? We adopt a specific formalism that accomodates a subset of the constructs found in regulation. We generalize from the formalized constructs to other constructs. Some of these generalizations may need revision in the future.

We assume that sentences in regulation are translated to statements in logic of the form:

$$(\mathrm{id})\ \varphi(x_1, ..., x_n) \mapsto \psi(x_1, ..., x_n)$$

where, "id" is an identifier, φ is *the precondition*, ψ is *the postcondition*, and $x_1, ..., x_n$ are free variables. The distinction between pre and postconditions has been adopted by most logics for regulation, to accomodate exceptions to laws (Sergot et al., 1986; Makinson and van der Torre, 2000; Governatori et al., 2006). The pre and postconditions are expressed in a modal logic that we designed in prior work (Dinesh et al., 2011). In describing the logical form, we will sketch how the logical form can be mapped to logic. But, we do not assume that the reader has a detailed understanding of the logic.

Given the assumptions about the logic, our goal is to transform a regulatory sentence into a structure that lets us determine: (I) the constituents of a sentence that contribute to the pre/postcondition, and (II) the scope of operators in the pre/postcondition. The structures that we use are called *abstract syntax trees* (ASTs), which can be understood as a restricted kind of logical form for regulatory texts.

1.2 Contributions and Outline

In this paper, we focus on the problem of computing the AST given a (kind of) parse tree for a sentence. The main step is is to *order or rank* scope-taking operators. A learning model for ranking is adapted for this ordering. We design features by studying the problem of comparing the scope of one operator to another. The pairwise scope comparisons are then used to compute ASTs, with an F-score of 90.6% on the set of ordering decisons.

The rest of this paper is organized as follows. We define ASTs using an example in Section 2, and setup the learning problem in Section 3. We then describe the corpus using statistics about operators in Section 4. In Section 5, we describe experiments on comparing the scope of an operator to another. We use the pairwise scope comparisons, in Section 6 to comput the AST. We discuss related work in Section 7 and conclude in Section 8.

2 Abstract Syntax Trees

We describe *abstract syntax trees* (ASTs) using an example from CFR Section 610.11:

(1) A general safety test for the detection of extraneous toxic contaminants shall be performed on biological products intended for administration to humans.

We discuss the translation in logic and the AST for the fragment of (1) that appears in black. In order to keep figures to a manageable size, we restrict attention to fragments of sentences, by graying out portions. The term AST is borrowed from compilers (Aho et al., 1986), where it is used as an intermediate step in the semantic interpretation of programs.
Translation in Logic: The sentence (1) is formally expressed as:

$$(1)\ \mathrm{bio\_prod}(x) \mapsto \mathcal{O}_{m(x)}(\exists y : \mathrm{test}(y) \wedge \psi(x, y))$$

where, $\psi(x, y) = \mathrm{gensaf}(y) \wedge \mathrm{ag}(y, m(x)) \wedge \mathrm{ob}(y, x)$

The predicates and function symbols are read as follows. $\mathrm{bio\_prod}(x)$ - "x is a biological product". $m(x)$ denotes the manufacturer of x. The modal operator $\mathcal{O}$ stands for "obligation". $\mathrm{test(y)}$ - "y is a test (event)". $\mathrm{gensaf}(y)$ - "y is a general safety procedure". $\mathrm{ag}(y, m(x))$ - "the agent of y is $m(x)$", and $\mathrm{ob}(y, x)$ - "the object of the event y is x". The formalized version of the law is read as follows: "If x is a biological product, then the manufacturer $m(x)$ is required/obligated to perform a general safety test y which has x as its object". We refer the reader to (Dinesh et al., 2011) for details on the logic.

The distinction between pre and postconditions is a non-trivial assumption. As with all logic-programming formalisms, only free variables are "shared" between pre and postconditons. This implies that all existential quantification, modals, and negation appear within the pre or postcondition. In the example above, the existential quantifier ($\exists y$) and the modal ($\mathcal{O}$) appear within the postcondition.

Abstract Syntax Tree: The AST for (1) is shown in Figure 1. The main nodes of interest are the internal nodes labeled $\boxed{\lambda}$. An internal node with $n + 1$ children corresponds to an *n-ary operator*. The first child of the internal node is the operator. Operators are labeled with a part-of-speech tag, for example, "D" for determiner, "M" for modal, and "O" for other. The remaining n children are its arguments. We use the term *nuclear scope* to refer to the last (n^{th}) argument of the operator, and the term *restrictor* to refer to any other argument. We borrow these terms from the literature on quantifier scope for determiners (Heim and Kratzer, 1998, Chapter 7).

For example, the phrase "general safety test" is in the restrictor of the operator A, and the variable $\boxed{y}$ is in its nuclear scope. The modal shall is a unary operator, and doesn't have a restrictor. Non-unary operators bind the variable displayed on the internal node. The variable $\boxed{y}$ is bound by the operator A.

Implicit operators are inserted when there is no overt word or phrase. In Figure 1, the implicit operators are underlined. The generic noun phrase "biological products" is associated with the implicit determiner all. Similarly, we use the implicit operator Post to mark the position of the postcondition.

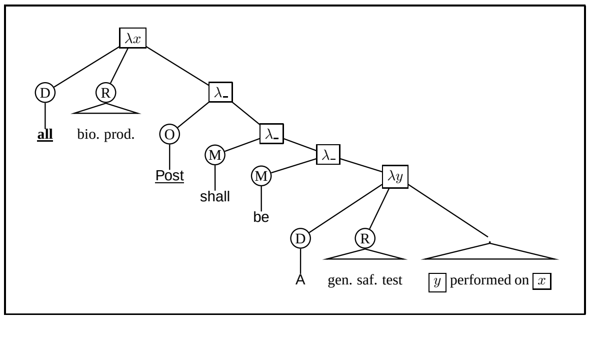

Figure 1: Example of an abstract syntax tree (AST).

We conclude this section with some notation for describing ASTs. Given an AST for a sentences, we say that an operator o_i *scopes over* o_j, denoted $o_i \gg o_j$, if o_j appears in the nuclear scope of o_i. For example, in Figure 1, we have all $\gg$ Post, all $\gg$ shall, all $\gg$ A, Post $\gg$ A, and shall $\gg$ A. In addition, we say that *the restrictor of* o_i *scopes over* o_j, denoted $R(o_i) \gg o_j$, if o_j appears in the restrictor of o_i. Such configurations occur with PP-modification of NPs, and we discuss examples in later sections.

3 Computing ASTs – Overview

In this section, we give an overview of our approach to computing ASTs. We will assume as given a Processed Parse Tree (PPT) of a sentence, with the operators and their restrictors identified. An example is discussed in Section 3.1. Given such a PPT, the AST is computed in two steps: (1) finding the preterminal at which an operator takes scope, and (2) ordering the operators associated with a preterminal. We describe the second step in Section 3.2, and then briefly outline the first step in Section 3.3. The steps are described in reverse order, because in most cases, the operators associated with a preterminal are determined directly by syntactic attachment.

3.1 Processed Parse Trees

We compute ASTs from processed parse trees (PPTs) of sentences. Figure 2 gives the PPT corresponding to the AST in Figure 1.

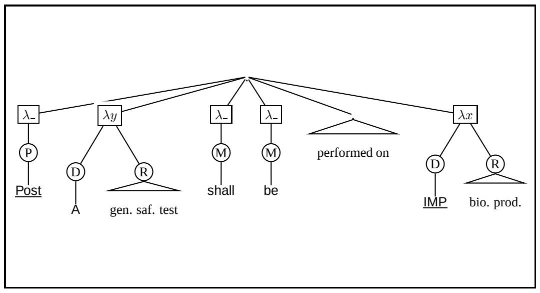

Figure 2: Processed parse tree (PPT) for (1).

A PPT provides the set of operators in a sentence, associated with their restrictors. For example, the determiner "a" has the restrictor *general safety test*. The phrase *biological products* has no explicit determiner associated with it, and the corresponding operator in the PPT is labeled "IMP" for *implicit*. In addition, the postcondition marker "Post" is also identified. Except for the postcondition marker, annotator-specified implicit operators are not given in the PPT.

There are two main types of nodes in the PPT – *operators* and *preterminals*. The nodes labeled with the symbol λ, e.g., $\boxed{\lambda\_}$ and $\boxed{\lambda x}$, correspond to operators. The root of the PPT and the restrictors of the operators, are the preterminals. Based on this example, it may seem that a sentence just has a list of operators. While this is true of example (1), embedded operators arise, for example, in the context of PP-modification of NPs and relative clauses. We will discuss an example in Section 3.3.

In this work, the PPTs are obtained by removing all scope decisions from the AST. To a first approximation, we start by removing all operators from the AST, and then, replace the corresponding variables by the operators. Implicit unary operators (such as the postcondition marker) are placed at the start of the preterminal.

It is worthwhile to consider whether it is reasonable to assume PPTs as given. We believe that this assumption is (slightly) stronger than assuming perfect parse trees. Although the PPT leaves certain chunks of the sentence unprocessed, in most cases, the unprocessed chunks correspond to base NPs. The main additional piece of information is the existence of a postcondition marker for each main clause of a sentence. We believe that computation of PPTs is better seen as a problem of syntax rather than scope, and we set it aside to future work. Our focus here is on converting a PPT to an AST.

3.2 Ordering Operators

The problem of learning to order a set of items is not new. Cohen et al. (1998) give a learning theoretic perspective, and Liu (2009) surveys information retrieval applications. The approach that we use can be seen as a probabilistic version of the boosting approach developed by Cohen et al. (1998). We explain the step of ordering operators, by revisiting the example of the general safety test, from Section 2.

Given the PPT in Figure 2, we compute the AST in Figure 1 by *ordering* or *ranking* the operators. For example, we need to determine that the implicit determiner associated with *biological products* is universal, and hence, we have $\underline{\mathsf{IMP}} \gg \underline{\mathsf{Post}}$. However, the determiner "A" associated with *general safety test* is existential, and hence, we have $\underline{\mathsf{Post}} \gg \mathsf{A}$.

We now develop some notation to describe the scopal ordering of operators. A PPT τ is viewed as a set of preterminal nodes, and we will write – (a) $\mathfrak{p} \in \tau$ to denote that $\mathfrak{p}$ occurs in τ, and (b) $|\tau|$ to denote the number of preterminals in τ. A preterminal $\mathfrak{p}$ is viewed as an ordered set of operators $\mathfrak{p} = (\mathsf{o}^1, ..., \mathsf{o}^{|\mathfrak{p}|})$. For example, in Figure 2, the root preterminal $\mathfrak{p}$ has $|\mathfrak{p}| = 5$, and the operators $\mathsf{o}^1 = \underline{\mathsf{Post}}$, $\mathsf{o}^2 = \mathsf{A}$, $\mathsf{o}^3 = \mathsf{shall}$, and so on.

An AST α contains a ranking of operators associated with each preterminal, denoted $\mathfrak{r}_\alpha(\mathfrak{p})$. The ranks of operators are denoted by subscripts. Let $\mathfrak{p} = (\mathsf{o}^1, ..., \mathsf{o}^5)$ be the root preterminal of the PPT in Figure 2. The ranking associated with the AST in Figure 1 is given by $\mathfrak{r}_\alpha(\mathfrak{p}) = (\mathsf{o}^1_2, \mathsf{o}^2_5, \mathsf{o}^3_3, \mathsf{o}^4_4, \mathsf{o}^5_1)$. For example, $\mathsf{o}^2_5 = \mathsf{A}$ denotes that the determiner "A" appears second in the surface order (Figure 2) and fifth or lowest in the scope order (Figure 1). Similarly, $\mathsf{o}^5_1 = \underline{\mathsf{IMP}}$ denotes that the implicit determiner appears fifth or last in the surface order (Figure 2) and first or highest in the scope order (Figure 1). Note that the subscript suffices to identify the position of an operator in the AST.

Model: We now describe the learning model for ordering operators. Given a PPT τ, let $\mathcal{A}(\tau)$ be the set of all possible ASTs. Our goal is to find the AST which has the highest probability given the PPT:

$$\alpha^* = \arg \max_{\alpha \in \mathcal{A}(\tau)} P(\alpha|\tau)$$

The conditional probability of an AST is defined as:

$$P(\alpha|\tau) = \prod_{\mathfrak{p} \in \tau} P(\mathfrak{r}_\alpha(\mathfrak{p})|\tau)$$

$$P(\mathfrak{r}_\alpha(\mathfrak{p})|\tau) = \prod_{i=1}^{|\mathfrak{p}|-1} \prod_{j=i+1}^{|\mathfrak{p}|} P(\mathsf{o}_\mathsf{i} \gg \mathsf{o}_\mathsf{j}|\tau)$$

In other words, $P(\alpha|\tau)$ is modeled as the product of the probabilities of the ranking of each preterminal, which is in turn expressed as the product of the probabilities of the pairwise ordering decisions. The model falls under the class of pairwise ranking approaches (Liu, 2009). We will consider the problem of estimating the probabilities in Section 5, and the problem of searching for the best AST in Section 6.

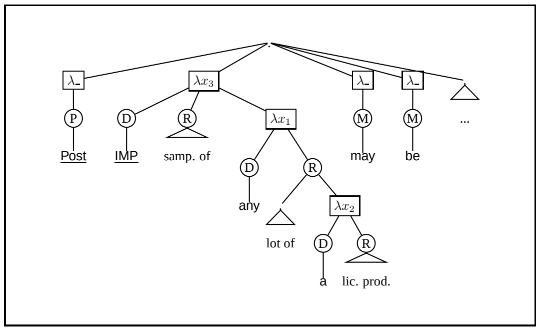

Figure 3: PPT for (2)

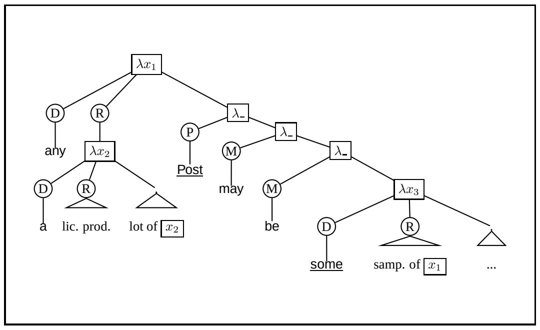

Figure 4: AST for (2)

3.3 Finding the Scope Preterminal

In the example that we discussed in the previous section, there were no embedded operators, i.e., an operator or its variable located in the restrictor of another. An embedded operator can either – (a) take scope within the restrictor of the embedding operator, or (b) outscope the embedding operator. To account for the second case, we need to determine whether it is appropriate to lift an embedded operator to a higher preterminal than the one to which it is associated syntactically.

We discuss an example of *inverse linking* (Larson, 1985) to illustrate the problem. Consider the following sentence:

(2) Samples of any lot of a licensed product, except for radioactive biological products, together with the protocols showing results of applicable tests, may at any time be required to be sent to the Director, Center for Biologics Evaluation and Research.

The PPT and AST for (2) are shown in Figures 3 and 4 respectively. Consider the noun phrase "IMP samples of any lot of a licensed product" in the

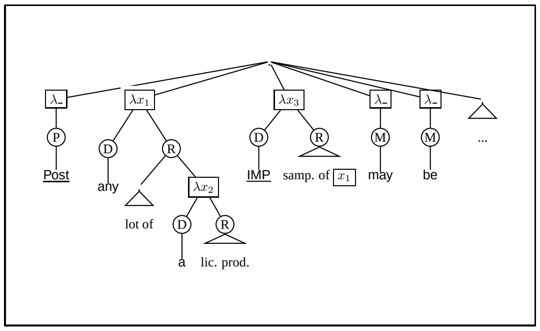

Figure 5: Second PPT for (2), obtained from the PPT in Figure 3, by raising any to the root preterminal.

PPT. The implicit determiner IMP in the PPT is interpreted as the existential determiner some in the AST. The three operators are related as follows in the AST: any $\gg$ some and R(any) $\gg$ a, i.e., any outscopes the implicit determiner, and a appears in the restrictor of any. Observe that the variables $\boxed{x_1}$ and $\boxed{x_2}$, which are associated with any and a, appear in the restrictors of some and any respectively. As a result, in the PPT, in Figure 3, any and a appear in the restrictor of IMP and any. The PPT provides a standard parse of PP-modification of NPs.

The important feature of this example is that the determiner "any" is syntactically embedded in the restrictor of IMP in the PPT (Figure 4), but it outscopes the implicit determiner in the AST (Figure 3). As a result, the PPT in Figure 3 cannot be converted to the AST in Figure 4 simply by ranking sibling operators (as we did in the previous section).

To handle such cases, we convert the PPT in Figure 3 to a second PPT (shown in Figure 5). The only allowed operation during this conversion is to raise an embedded operator to a higher preterminal. The PPT in Figure 5 is obtained by raising any to the root preterminal, making it a sibling of the implicit determiner IMP in the PPT in Figure 5. This second PPT can be converted to the AST by reordering sibling operators. The learning model used for this step is similar to the one used to order operators, and in the interests of space, we omit the details.

4 Brief Overview of the Corpus

We have annotated 195 sentences from the FDA CFR Section 610 with ASTs. The operators are divided into the following types – determiners (e.g., every, a, at least), modal auxiliaries (e.g., must, be), VP modifiers (e.g., if, for, after), negation and coordinating conjunctions (e.g., and, but, or). The majority of the corpus was annotated by a single annotator. However, to estimate inter-annotator agreement, a set of 32 sentences was annotated by a second annotator. In this section, we restrict ourselves to presenting statistics that highlight part of the guidelines and motivate the features that we use to order operators. An example-based justification of guidelines, and a discussion of inter-annotator agreement can be found in (Dinesh, 2010).

De Re vs De Dicto: We narrow our focus to one part of the annotation, the *de re* vs *de dicto* distinction. Informally, operators with *de re* scope occur in the precondition of the logical translation of a sentence, while those with *de dicto* scope occur in the postcondition. This distinction is of key importance in the application of conformance checking, as it helps determine the facts that need to be provided by an organization (*de re*), and the actions that an organization is required to take (*de dicto*).

For simplicity, we further restrict attention to operators that are siblings of the postcondition in the AST, and ignore the operators embedded in prepositional phrases and clauses, for example. A (main clause) operator o is said to have *de re* scope iff it outscopes the postcondition marker ($o \gg \underline{\text{Post}}$). Otherwise, the operator is said to have *de dicto* scope ($\underline{\text{Post}} \gg o$). In the example of the general safety test from Section 2, the implicit determiner associated with "biological products" has *de re* scope, while all other operators in the sentence have *de dicto* scope.

Operator Type	Number of Instances	De Re Scope Percentage
Determiner	277	59.9%
Modal Aux	268	0%
VP Modifier	132	68.2%
CC	36	22.2%
Neg	33	0%
Other	74	17.6%

Table 1: De Re scope distribution. An operator has *de re* scope iff it outscopes the postcondition marker.

Table 1 shows the percentage of each type of operator that has *de re* scope. Modal auxiliaries and negation are umambigous to this distinction, and always have *de dicto* scope. Note that a type of operator with 50% occuring *de re* is ambiguous, while 0% or 100% are unambiguous. Thus, from Table 1, we can conclude that determiners, and VP modifiers are the most ambiguous types. And, more features are needed to disambiguate them.

Determiner Type	Number of Instances	De Re Scope Percentage
Universal	74	100%
Existential	12	0%
Ambiguous	50	28%
Deictic	127	53.5%
Other	14	35.7%

Table 2: De Re scope distribution for determiners.

Determiners: We divide the determiners into the following subtypes: universal/generic (e.g., every, all), existential (some), ambiguous (e.g., a, an), deictic (e.g., the, those), and other (e.g., at least, at most). The guidelines for annotation were as follows – (a) universal determiners have *de re* scope, (b) existential determiners have *de dicto* scope, and (c) for other determiners, the annotator needs to decide whether a particular use is interpreted existentially or universally. Table 2 shows *de re* scope distribution for each of these subtypes. As expected, universal and existential determiners are unambiguous, while ambiguous and deictic determiners show more variety. For example, the deictic determiner the can refer to a specific entity ("the FDA") or have a universal interpretation ("the products").

Thus, to disambiguate between *de re* and *de dicto* interpretations for determiners, we need two types of features – (1) Features to predict whether ambiguous and deictic determiners are universal or not, and (2) Features to determine the type of implicit determiners. In Table 2, we assume that the type of implicit determiners are given. This assumption is unrealistic. Rather, we need to predict the type of such determiners, during the computation of the AST.

VP Modifier Type	Number of Instances	De Re Scope Percentage
Temporal and Conditional	73	100%
Purpose	8	0%
References to Laws	33	0.9%
Other	29	65.5%

Table 3: De Re scope distribution for VP modifiers.

VP Modifiers: We divide the VP modifiers into the following subtypes: temporal and conditional (e.g., after, if), purpose (for), references to laws (which are a special type of modifier in the legal domain, e.g., "as specified in paragraph (c)"), and other (e.g., regardless, notwithstanding). Table 3 shows the percentage of each subtype of modifier that has *de re* scope. Following the guidelines for annotation, the temporal and conditional modifiers are always *de re*, the purpose modifiers and modifiers conveying references to laws are always *de dicto*.

5 Comparing the Scope of Operators

We now consider a subproblem in computing the AST – comparing the scope of pairs of operators. In Section 6, we will use the classifiers that perform comparisons, to compute the AST. All experiments in this section use the MAXENT implentation from the MALLET toolkit (McCallum, 2002). We begin by revisiting *de re*-*de dicto* distinction from Section 4. Then, we generalize to other comparisons.

De Re vs De Dicto: The (binary) classification problem is as follows. Our observations are triples $x = (\mathsf{o}, \mathsf{o}', \tau)$ are such that there is a preterminal $\mathfrak{p} \in \tau$, $\{\mathsf{o}, \mathsf{o}'\} \subseteq \mathfrak{p}$, and $\mathsf{o}' = \underline{\mathsf{Post}}$. In other words, we are considering operators (o) that are siblings of the postcondition marker (o'). An observation has the label 1 if $\mathsf{o} \gg \mathsf{o}'$ (*de re* scope), and a label of 0 otherwise (*de dicto* scope).

Features: We use the following (classes of) features for an observation $x = (\mathsf{o}, \mathsf{o}', \tau)$:

- TYPE - The type and subtype of the operator. We use the subtypes from Section 4 only for explicit operators.

- PRE-VERB - Tracks whether o and o' appear before or after the main verb of the sentence.

- PRE-VERB + PERF - Conjunction of the previous feature with whether the main verb is *perform*. The verb *perform* is frequent in the CFR, and its subject is typically given *de dicto* scope, as it is the main predicate of the sentence.

- POS - The part-of-speech of the head word. For example, for the noun phrase *biological products*, the head word is *products*, and the POS is NNS (plural common noun). And, this POS tag may indicate a generic/universal interpretation.

	Count	MAJORITY	TYPE	ALL
All	823	66.2%	84.1%	89.2%
No MD	522	53.2%	74.9%	83.7%
DT	277	59.9%	62.9%	81.2%
Imp. DT	100	69%	-	76%

Table 4: De Re vs De Dicto classification. Average accuracies over 10-fold cross-validation. The rows describe the subset of observations considered, and the columns describe the subset of features used.

Experiments: We evaluate the features by performing 10-fold cross-validation. The results are summarized in Table 4. The rows describe the subset of observations used. "All" includes all observations, "No MD" excludes the modal auxiliaries, "DT" includes only the determiners, and "Imp. DT" includes only implicit determiners. The columns describe the features used. MAJORITY is the majority baseline, i.e., the accuracy obtained by predicting the most frequent class or the majority class. The majority class is *de dicto* when all operators are considered (the first row), and *de re* in all other rows. The TYPE column gives the accuracy when only the type and subtypes are used as features. This column does not apply to implicit determiners, as the subtype information is unavailable. And, finally, the ALL column gives the accuracy when all features are used.

From Table 4, we can conclude that the TYPE feature is useful in making the *de re*-*de dicto* distinction, and further gains are obtained by using ALL features. The most dramatic improvement is for determiners, and indeed, our features were designed for this case. However, the performance gains are not very high for implicit determiners, and further investigation is needed.

Next, we apply the features to more general operator comparisons. The first row of Table 5 considers observations $x = (\mathsf{o}, \mathsf{o}', \tau)$, where o and o' are siblings, and predicts whether $\mathsf{o} \gg \mathsf{o}'$. The second row considers observations where o' is embedded syntactically within o, and predicts whether R(o) $\gg \mathsf{o}'$. In other words, the problem is to determine whether a syntactically embedded operator remains scopally embedded, or whether it has inverse scope (see Section 3.3).

	Count	MAJORITY	TYPE	ALL
Siblings	2793	76.1%	83.3%	87.5%
Embedded	5081	95%	95.3%	96.4%

Table 5: Ordering siblings and embedded operators. Average accuracies over 10-fold cross-validation. The columns describe the subset of features used.

6 From Operator Comparisons to ASTs

We now consider the problem of computing the AST given the classifiers for comparing operators. Section 6.1 describes the algorithms used. In Section 6.2, we develop metrics to evaluate the computation of ASTs. We conclude, in Section 6.3, by evaluating different algorithms using the metrics.

6.1 Algorithms

We begin by discussing the intractability of the problem of ranking or ordering operators. Then, we sketch the search heuristics used.

Intractability: The decision version of the ranking problem is NP-complete. A similar result is established by Cohen et al. (1998) in the context of a boosting approach to ranking.

Theorem 1. *The following problem is NP-complete:*
Input: *A PPT τ, a preterminal $\mathfrak{p} \in \tau$, probabilities $P(\mathsf{o}^{\mathsf{i}} \gg \mathsf{o}^{\mathsf{j}}|\tau)$, and $c \in [0, 1]$*
Output: *Yes, if there is an ordering $\mathfrak{r}$ such that $P(\mathfrak{r}(\mathfrak{p})|\tau) \geq c$*

The proof is by reduction from ACYCLIC SUBGRAPH (Karp, 1972) – finding a subgraph which is acyclic and has at least k edges.

Heuristics: To order operators, we use a beam search procedure. Each search state consists preterminal, in which the first i ranks have been assigned to operators. We then search over next states by assigning the rank $i + 1$ to one of the remaining operators. We used a beam size of 10^4 in our experiments. In most cases, the number of operators per preterminal is less than 7. As a result, the total number of possible orderings is typically less than 7!, and a beam size of 10^4 is sufficient to compute an exact ordering. In other words, due to the size restrictions, in most cases, beam search is equivalent to exact (exhaustive) search.

To handle embedded operators, we use a simple greedy heuristic. We enumerate the operators in the initial PPT, corresponding to an in-order traversal. For each operator, we attach it to the most likely ancestor, given the attachment decisions for the previous operators. This heuristic is optimal for the case where the depth of embedding is at most 1, which is the common case.

6.2 Metrics

In this section, we describe metrics used to evaluate the computation of ASTs. Let τ be the initial PPT, α the correct AST, and α^* the computed AST. We define accuracy at various levels.

The simplest metric is to define accuracy at *the level of ASTs*, i.e., by computing the fraction of cases for which $\alpha = \alpha^*$. However, this metric is harsh, in the sense that it does not give algorithms partial credit for getting a portion of the AST correct.

The next possible metric is to define accuracy at *the level of preterminals*. Let $\mathfrak{p}$ be a preterminal. Note that τ, α and α^* share the same set of preterminals, but may associate different operators with them. We say that $\mathfrak{p}$ is correct in α^*, if it is associated with the same set of operators as in α, and for all $\{\mathsf{o}, \mathsf{o}'\} \subseteq \mathfrak{p}$, we have $\mathsf{o} \gg \mathsf{o}'$ w.r.t. α^* iff $\mathsf{o} \gg \mathsf{o}'$ w.r.t. α. In other words, the preterminals are identical, both in terms of the set of operators and the ordering between pairs of operators. While preterminal-level accuracy gives partial credit, it is still a little harsh, in the sense that an algorithm which makes one ordering mistake at a preterminal is penalized the same as an algorithm which makes multiple mistakes.

Finally, we consider metrics to define accuracy at *the level of pairs of operators*. Let $\mathfrak{p}$ be a preterminal. The set $\mathrm{Pairs}(\mathfrak{p}, \alpha)$ consists of pairs of operators $(\mathsf{o}, \mathsf{o}')$ such that o and o' are both associated with $\mathfrak{p}$ in α, and $\mathsf{o} = \mathsf{o}'$ or $\mathsf{o} \gg \mathsf{o}'$. The set $\mathrm{Pairs}(\mathfrak{p}, \alpha^*)$ is defined similarly using α^* instead of α. Given the sets $\mathrm{Pairs}(\mathfrak{p}, \alpha)$ and $\mathrm{Pairs}(\mathfrak{p}, \alpha^*)$, precision, recall, and f-score are defined in the usual way. We leave the details to the reader.

6.3 Results

We evaluate the following algorithms:

1. No Embedding – The AST is computed purely by reordering operators within a preterminal in the PPT.

(a) SURFACE – No reordering is performed, i.e., the order of operators in the AST respects the surface order

(b) TYPE – Using only type and subtype information for the operators

(c) ALL – Using all the features described in Section 5

2. ALL+ – The initial PPT is transformed into a second PPT before reordering (as described in Section 3.3). All features are used.

	Prec.	Rec.	F	$\mathfrak{p}$	α
SURF.	86.9%	82.7%	84.6%	81%	4.2%
TYPE	90.4%	86%	88.1%	83.6%	24.7%
ALL	**92%**	87.6%	89.8%	85.1%	33.5%
ALL+	91.9%	**89.4%**	**90.6%**	**85.9%**	**36.2%**

Table 6: Performance of the algorithms in computing the ASTs. Averaged over 10-fold cross-validation. 195 ASTs in total, an average of 8.6 preterminals per AST, and 1.8 operators per preterminal.

Table 6 summarizes the performance of the algorithms, under the various metrics. The accuracies are averaged over 10-fold cross-validation. A total of 195 ASTs are used. The average number of preterminals per AST is 8.6, with an average of 1.8 operators per preterminal. The best number under each metric is shown in bold-face. By adding features, we improve the precision from 86.9% to 90.4% to 92% in moving from SURFACE to TYPE to ALL. By handling embedded operators, we improve the recall from 87.6% to 89.4% in moving from ALL to ALL+. As we saw in Section 5, in 95% of the cases, the embedded operators respects syntactic scope, and as a result, we obtain only modest gains from handling embedded operators.

The reader may feel that the F-score of 90.6% is quite high given the size of our training data. This score is inflated by inclusion of reflexive pairs, of the form (o, o). Such pairs are included for the following (technical) reasons. The algorithm that handles embedded operators (ALL+) usually raises them from a single operator node (as in Figure 3) to a multi-operator node (as in Figure 5). If it makes an incorrect decision to raise an operator it takes a precision hit, at the multi-operator node (because it has some false positives). By contrast, an algorithm loses precision for failing to correctly raise, only when we encounter the single operator node.

For these reasons, it is better to consider the relative improvement in F-score over the baseline. The relative improvement of ALL+ over SURFACE in terms of F-score is 36.6%. We believe that the preterminal-level accuracy is more indicative in an absolute sense. Furthermore, when we restrict attention to those preterminals with two or more operators in the PPT, the accuracy of ALL+ is 69.4%.

7 Related Work

ASTs can be seen as a middle ground between two lines of research in translating sentences to logic.

At one end of the spectrum, we have methods that achieve good accuracy on restricted texts. The two main corpora that have been considered are the GEOQUERY corpus (Thompson et al., 1997) and the ATIS-3 corpus (Dahl et al., 1994). The GEOQUERY corpus consists of queries to a geographical database. The queries were collected from students participating in a study and the average sentence length is 8 words. The ATIS corpus is collected from subjects' interaction with a database of flight information, using spoken natural language. The utterances have be transcribed, and the average sentence length is 10 words (Berant et al., 2007). Algorithms, which achieve good accuracy, have been developed to compute the logical translation for these queries (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Zettlemoyer and Collins, 2009). The annotated sentences in the FDA CFR Section 610.40 are longer (about 30 words on average), and contain modalities which are not present in these corpora.

At the other end of the spectrum, Bos et al. (Bos et al., 2004) have developed a broad-coverage parser to translate sentences to a logic based on discourse representation theory. Here, there is no direct method to evaluate the correctness of the translation. However, indirect evaluations are possible, for example, by studying improvement in textual entailment tasks.

To summarize, there are techniques that either produce an accurate translation for sentences in a limited domain, or produce some translation for sentences in a broader range of texts. ASTs offer a middle ground in two ways. First, we focus on regulatory texts which are less restricted than the database queries in the GEOQUERY and ATIS corpora, but do not exhibit anaphoric phenomenon found in genres, such as, newspaper text. In (Dinesh et al., 2007), we discuss lexical statistics that show significant differences in the distribution of anaphoric items in the CFR and Wall Street Journal (WSJ) corpora. For example, the frequency of pronouns and anaphoric discourse connectives is significantly lower in the CFR than in the WSJ. Instead, the CFR has an idiosyncratic mechanism for referring to sentences, using phrases such as "except as specified in paragraph (c) and (d)". A question of interest is whether the GEOQUERY and ATIS corpora show similar peculiarities in terms of anaphora. The second difference between our approach and others is that we do not attempt to translate all the way to logic. The level of logical form lets us obtain a direct evaluation, while leaving open the design of parts of the logic.

8 Conclusions

We described experiments on a modest-sized corpus of regulatory sentences, annotated with a novel variant of logical form, called *abstract syntax trees* (ASTs). An example from the corpus was presented in Section 2 and some statistics, describing the corpus, were discussed in Section 4. In Sections 3, 5, and 6, we developed and tested algorithms to convert a processed parse tree (PPT) to an AST. The main step in this conversion was to rank or order the operators at a preterminal. We presented a probabilistic model for ranking, investigated the design of features, and developed search heuristics. The best algorithm, which uses all features and handles embedded operators, achieves an F-score of 90.6%.

An important direction for further inquiry is in the design of better features. Various types of features have been proposed for the scopal ordering of determiners. Examples include syntactic features (Ioup, 1975; Reinhart, 1983), such as position and voice, semantic features (Grimshaw, 1990; Jackendoff, 1972), such as thematic roles. More recently, Srinivasan and Yates (2009) showed how pragmatic information, for example "there are more people than cities", can be leveraged for scope disambiguation. We experimented with lexico-syntactic features in this work, and leave an investigation of semantic and pragmatic features to future work.

Acknowledgements

We thank Claire Cardie, Steve Kimbrough, Annie Louis, Fernando Pereira, Emily Pitler, Oleg Sokolsky, and the anonymous reviewers for helpful comments on earlier versions of this paper.

References

A. Aho, R. Sethi, and J. Ullman. 1986. *Compilers: Principles, Techniques, and Tools*. Addison-Wessley.

R. J. Anderson. 1996. A security policy model for clincial information systems. In *Proceedings of the IEEE Symposium on Security and Privacy*.

A. Barth, A. Dutta, J. C. Mitchell, and H. Nissenbaum. 2006. Privacy and contextual integrity: Framework and applications. In *Proceedings IEEE Symposium on Security and Privacy*.

J. Berant, Y. Gross, M. Mussel, B. Sandbank, E. Ruppin, and S. Edelman. 2007. Boosting unsupervised grammar induction by splitting complex sentences on function words. In *Proceedings of the Boston University Conference on Language Development*.

J. Bos, S. Clark, M. Steedman, J. R. Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING*.

W. W. Cohen, R. E. Schapire, and Y. Singer. 1998. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270.

D. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. Expanding the scope of the ATIS task: the ATIS-3 corpus. In *Proceedings of the ARPA HLT Workshop*.

N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky. 2007. Logic-based regulatory conformance checking. In *Proceedings of the 14th Monterey Workshop*.

N. Dinesh, A. Joshi, I. Lee, and O. Sokolsky. 2011. Permission to speak: A logic for access control and conformance. *Journal of Logic and Algebraic Programming*, 80(1):50–74.

N. Dinesh. 2010. *Regulatory Conformance Checking: Logic and Logical Form*. Ph.D. thesis, University of Pennsylvania.

G. Governatori, Z. Milosevic, and S. Sadiq. 2006. Compliance checking between business processes and business contracts. In *10th International Enterprise Distributed Object Computing Conference (EDOC)*.

J. Grimshaw. 1990. *Argument Structure*. MIT Press.

B. Grosof, Y. Labrou, and H. Y. Chan. 1999. A declarative approache to business rules in contracts: Courteous logic programs in xml. In *ACM Conference on Electronic Commerce*.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell.

G. Ioup. 1975. Some universals for quantifier scope. *Syntax and Semantics*, 4:37–58.

R. Jackendoff. 1972. *Semantic Interpretation in Generative Grammar*. MIT Press.

A. J. I. Jones and M. J. Sergot. 1992. Formal specification of security requirements using the theory of normative positions. In *European Symposium on Reasearch in Computer Security (ESORICS)*.

R. M. Karp. 1972. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press.

H. S. Kurtzman and M. C. MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition*, 48:243–279.

R. K. Larson. 1985. Quantifying to np. Manuscript, MIT.

T. Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3).

D. Makinson and L. van der Torre. 2000. Input/output logics. *Journal of Philosophical Logic*, 29:383–408.

R. May. 1985. *Logical Form: Its structure and derivation*. MIT Press.

A. McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

T. Reinhart. 1983. *Anaphora and Semantic Interpretation*. Croom Helm.

M.J. Sergot, F.Sadri, R.A. Kowalski, F.Kriwaczek, P.Hammond, and H.T. Cory. 1986. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–86.

P. Srinivasan and A. Yates. 2009. Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

C. A. Thompson, R. J. Mooney, and L. R. Tang. 1997. Learning to parse natural language database queries into logical form. In *Proceedings of the Workshop on Automata Induction, Grammatical Inference and Language Acquisition*.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*.

L. S. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.