CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank

Julia Hockenmaier[*]
University of Pennsylvania

Mark Steedman[**]
University of Edinburgh

This article presents an algorithm for translating the Penn Treebank into a corpus of Combinatory Categorial Grammar (CCG) derivations augmented with local and long-range word–word dependencies. The resulting corpus, CCGbank, includes 99.4% of the sentences in the Penn Treebank. It is available from the Linguistic Data Consortium, and has been used to train wide-coverage statistical parsers that obtain state-of-the-art rates of dependency recovery.

In order to obtain linguistically adequate CCG analyses, and to eliminate noise and inconsistencies in the original annotation, an extensive analysis of the constructions and annotations in the Penn Treebank was called for, and a substantial number of changes to the Treebank were necessary. We discuss the implications of our findings for the extraction of other linguistically expressive grammars from the Treebank, and for the design of future treebanks.

1. Introduction

In order to understand a newspaper article, or any other piece of text, it is necessary to construct a representation of its meaning that is amenable to some form of inference. This requires a syntactic representation which is transparent to the underlying semantics, making the local and long-range dependencies between heads, arguments, and modifiers explicit. It also requires a grammar that has sufficient coverage to deal with the vocabulary and the full range of constructions that arise in free text, together with a parsing model that can identify the correct analysis among the many alternatives that such a wide-coverage grammar will generate even for the simplest sentences. Given our current machine learning techniques, such parsing models typically need to be trained on relatively large treebanks—that is, text corpora hand-labeled with detailed syntactic structures. Because such annotation requires linguistic expertise, and is therefore difficult to produce, we are currently limited to at most a few treebanks per language.

One of the largest and earliest such efforts is the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993; Marcus et al. 1994), which contains a one-million word

[*] Institute for Research in Cognitive Science, University of Pennsylvania, 3401 Walnut Street, Suite 400A, Philadelphia, PA 19104-6228, USA. E-mail: juliahr@cis.upenn.edu.

[**] School of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, UK. E-mail: steedman@inf.ed.ac.uk.

Submission received: 16 July 2005; revised submission received: 24 January 2007; accepted for publication: 21 February 2007.

subcorpus of *Wall Street Journal* text that has become the de facto standard training and test data for statistical parsers. Its annotation, which is based on generic phrase-structure grammar (with coindexed traces and other null elements indicating non-local dependencies) and function tags on nonterminal categories providing (a limited degree of) syntactic role information, is designed to facilitate the extraction of the underlying predicate–argument structure. Statistical parsing on the Penn Treebank has made great progress by focusing on the machine-learning or algorithmic aspects (Magerman 1994; Ratnaparkhi 1998; Collins 1999; Charniak 2000; Henderson 2004; McDonald, Crammer, and Pereira 2005). However, this has often resulted in parsing models and evaluation measures that are both based on reduced representations which simplify or ignore the linguistic information represented by function tags and null elements in the original Treebank. (One exception is Collins 1999, whose Model 2 includes a distinction between arguments and adjuncts, and whose Model 3 additionally captures *wh*-movement in relative clauses with a GPSG-like "slash-feature-passing" mechanism.)

The reasons for this shift away from linguistic adequacy are easy to trace. The very healthy turn towards quantitative evaluation interacts with the fact that just about every dimension of linguistic variation exhibits a Zipfian distribution, where a very small proportion of the available alternatives accounts for most of the data. This creates a temptation to concentrate on capturing the few high-frequency cases at the top end of the distribution, and to ignore the "long tail" of rare events such as non-local dependencies. Despite the fact that these occur in a large number of sentences, they affect only a small number of words, and have thus a small impact on overall dependency recovery.

Although there is now a sizable literature on trace and function-tag insertion algorithms (Blaheta and Charniak 2000; Johnson 2002; Campbell 2004), and integrated parsing with function tags or null elements (Dienes and Dubey 2003a, 2003b; Merlo and Musillo 2005; Gabbard, Kulick, and Marcus 2006), such approaches typically require additional pre- or postprocessing steps that are likely to add further noise and errors to the parser output. A completely integrated approach that is based on a syntactic representation which allows direct recovery of the underlying predicate–argument structure might therefore be preferable. Such representations are provided by grammar formalisms that are more expressive than simple phrase-structure grammar, like Lexical-Functional Grammar (LFG) (Kaplan and Bresnan 1982), Head-driven Phrase-Structure Grammar (HPSG) (Pollard and Sag 1994), Tree-Adjoining Grammar (TAG) (Joshi and Schabes 1992), Minimalist Program–related Grammars (Stabler 2004), or Combinatory Categorial Grammar (CCG) (Steedman 1996, 2000). However, until very recently, only handwritten grammars, which lack the wide coverage and robustness of Treebank parsers, were available for these formalisms (Butt et al. 1999; XTAG-group 1999; Copestake and Flickinger 2000; OpenCCG[1] [White and Baldridge 2003; White 2006]).

Because treebank annotation for individual formalisms is prohibitively expensive, there have been a number of efforts to extract TAGs, LFGs, and, more recently, HPSGs, from the Penn Treebank (Xia 1999; Chen and Vijay-Shanker 2000; Xia, Palmer, and Joshi 2000; Xia 2001; Cahill et al. 2002; Miyao, Ninomiya, and Tsujii 2004; O'Donovan et al. 2005; Shen and Joshi 2005; Chen, Bangalore, and Vijay-Shanker 2006). Statistical parsers that are trained on these TAG and HPSG corpora have been presented by Chiang (2000) and Miyao and Tsujii (2005), whereas the LFG parsing system of Cahill et al. (2004) uses

[1] Open CCG, the successor of Grok (Hockenmaier et al. 2004), is available from `http://openccg.sourceforge.net`.

a postprocessing step on the output of a Treebank parser to recover predicate–argument dependencies.

In this article we present an algorithmic method for obtaining a corpus of CCG derivations and dependency structures from the Penn Treebank, together with some observations that we believe carry wider implications for similar attempts with other grammar formalisms and corpora. Earlier versions of the resulting corpus, CCGbank, have already been used to build a number of wide-coverage statistical parsers (Clark, Hockenmaier, and Steedman 2002; Hockenmaier and Steedman 2002; Hockenmaier 2003b, 2003a; Clark and Curran 2004, 2007), which recover both local and long-range dependencies directly and in a single pass.

CCG is a linguistically expressive, but efficiently parseable, lexicalized grammar formalism that was specifically designed to provide a base-generative account of coordinate and relativized constructions like the following:

a. *pay HealthVest $5 million right away and additional amounts in the future* (1)

b. *the parched Franco years, the everyday poverty and stagnant atmosphere of which he described in brutally direct, vivid prose*

c. *Who is, and who should be, making the criminal law here?*

CCG directly captures the non-local dependencies involved in these and other constructions, including control and raising, via an enriched notion of syntactic types, without the need for syntactic movement, null elements, or traces. It also provides a "surface-compositional" syntax–semantics interface, in which monotonic rules of semantic composition are paired one-to-one with rules of syntactic composition. The corresponding predicate–argument structure or logical form can therefore be directly obtained from any derivation if the semantic interpretation of each lexical entry is known. In this article and in CCGbank, we approximate such semantic interpretations with dependency graphs that include most semantically relevant non-anaphoric local and long-range dependencies. Although certain decisions taken by the builders of the original Penn Treebank mean that the syntactic derivations that can be obtained from the Penn Treebank are not always semantically correct (as we will discuss), subsequent work by Bos et al. (2004) and Bos (2005) has demonstrated that the output of parsers trained on CCGbank can also be directly translated into logical forms such as Discourse Representation Theory structures (Kamp and Reyle 1993), which can then be used as input to a theorem prover in applications like question answering and textual entailment recognition.

Translating the Treebank into this more demanding formalism has revealed certain sources of noise and inconsistency in the original annotation that have had to be corrected in order to permit induction of a linguistically correct grammar. Because of this preprocessing, the dependency structures in CCGbank are likely to be more consistent than those extracted directly from the Treebank via heuristics such as those given by Magerman (1994) and Collins (1999), and therefore may also be of immediate use for dependency-based approaches. However, the structure of certain constructions, such as compound nouns or fragments, is deliberately underspecified in the Penn Treebank. Although we have attempted to semi-automatically restore the missing structure wherever possible, in many cases this would have required additional manual annotation, going beyond the scope of our project. We suspect that these properties of the original Treebank will affect any similar attempt to extract dependency structures or grammars for other expressive formalisms. The Penn Treebank is the earliest (and still the largest) corpus of its kind; we hope that our experiences will extend its useful life, and help in the design of future treebanks.

2. Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) was originally developed as a "near-context-free" theory of natural language grammar, with a very free definition of derivational structure adapted to the analysis of coordination and unbounded dependency without movement or deletion transformations. It has been successfully applied to the analysis of coordination, relative clauses and related constructions, intonation structure, binding and control, and quantifier scope alternation, in a number of languages—see Steedman and Baldridge (2006) for a recent review. Extensions of CCG to other languages and word-orders are discussed by Hoffman (1995), Kang (1995), Bozsahin (1998), Komagata (1999), Steedman (2000), Trechsel (2000), Baldridge (2002), and Çakıcı (2005). The derivations in CCGbank follow the analyses of Steedman (1996, 2000), except where noted.

2.1 Lexical Categories

Categorial Grammars are strongly *lexicalized*, in the sense that the grammar is entirely defined by a lexicon in which words (and other lexical items) are associated with one or more specific *categories* which completely define their syntactic behavior. The set of categories consists of basic categories (e.g., S, NP, PP) and complex categories of the form X/Y or X\Y, representing functors with (basic or complex) argument category Y and result category X. Functor categories of the form X/Y expect their argument Y to its right, whereas those of the form X\Y expect Y to their left.[2] These functor categories encode subcategorization information, that is, the number and directionality of expected arguments. English intransitive verbs and verb phrases have the category S\NP: they take a (subject) NP to their left as argument and yield a sentence. English transitive verbs have the category (S\NP)/NP: they take an (object) NP to their right to yield a verb phrase (S\NP), which in turn takes a (subject) NP to its left to form a sentence S. Each syntactic category also has a corresponding semantic interpretation (here given as a λ-expression). Hence, the lexical entry for ditransitive *give* can be written as follows:[3]

$$give := ((\mathtt{S}\backslash\mathtt{NP})/\mathtt{NP})/\mathtt{NP}{:}\lambda x\lambda y\lambda z.give'yxz \tag{2}$$

In our translation algorithm, we use simple word–word dependency structures to approximate the underlying semantic interpretation.

2.2 Derivations

A universal set of syntactic combinatory rules defines how constituents can be combined. All variants of categorial grammar since Ajdukiewicz (1935) and Bar-Hillel (1953) include function application, where a functor X/Y or X\Y is applied to an argument Y:

$$\begin{array}{llllll} \textit{Forward Application:} & \mathtt{X/Y}{:}\,f & \mathtt{Y}{:}\,a & \Rightarrow & \mathtt{X}{:}\,fa & (3)\\ \textit{Backward Application:} & \mathtt{Y}{:}\,a & \mathtt{X\backslash Y}{:}\,f & \Rightarrow & \mathtt{X}{:}\,fa & \end{array}$$

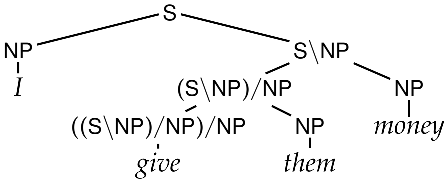
[2] This is the "result-leftmost" notation, which is easiest to mentally translate into the corresponding semantic type. There is an alternative "result-on-top" notation.

[3] This category embodies a lexicalized "Wrap" relation between the rightward NP arguments and the corresponding variables x and y in the semantic interpretation (Bach 1976; Dowty 1978; Jacobson 1992), reflecting the binding-theoretic relations between these arguments at the level of logical form rather than surface derivation.

These rules give rise to derivations like the following:[4]

$$\begin{array}{ccccl}
\textit{I} & \textit{give} & \textit{them} & \textit{money} & \\
\hline
\mathsf{NP}: I' & ((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP})/\mathsf{NP} : \lambda x\lambda y\lambda z.give'yxz & \mathsf{NP} : them' & \mathsf{NP} : money' & \\
\cline{2-3}
 & \multicolumn{2}{c}{(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda y\lambda z.give'y\,them'z} & & > \\
\cline{2-4}
 & \multicolumn{3}{c}{\mathsf{S}\backslash\mathsf{NP} : \lambda z.give'money'them'z} & > \\
\cline{1-3}
\multicolumn{3}{c}{\mathsf{S} : give'money'them'I'} & & < \\
\end{array} \tag{4}$$

This derivation is isomorphic to a traditional context-free derivation tree like the following (the semantics is omitted):



CCG additionally introduces a set of rule schemata based on the combinators of combinatory logic (Curry and Feys 1958), which enable succinct analyses of extraction and coordination constructions. It is a distinctive property of CCG that all syntactic rules are purely *type-driven*, unlike traditional structure-dependent transformations. Composition and substitution allow two functors to combine into another functor, whereas type-raising is a unary rule that exchanges the roles of functor and argument:

$$\begin{array}{lllll}
\textit{Forward Composition:} & \mathsf{X/Y}{:}f & \mathsf{Y/Z}{:}g & \Rightarrow_{\mathbf{B}} & \mathsf{X/Z}{:}\lambda z.f(gz) \\
\textit{Backward Composition:} & \mathsf{Y\backslash Z}{:}g & \mathsf{X\backslash Y}{:}f & \Rightarrow_{\mathbf{B}} & \mathsf{X\backslash Z}{:}\lambda z.f(gz) \\
\textit{Backward Crossed Composition:} & \mathsf{Y/Z}{:}g & \mathsf{X\backslash Y}{:}f & \Rightarrow_{\mathbf{B}} & \mathsf{X/Z}{:}\lambda z.f(gz) \\
\textit{Forward Type-raising:} & \mathsf{X}{:}a & & \Rightarrow_{\mathbf{T}} & \mathsf{T/(T\backslash X)}{:}\lambda f.fa \\
\textit{Backward Crossed Substitution:} & \mathsf{Y\backslash Z}{:}g & \mathsf{(X\backslash Y)\backslash Z}{:}f & \Rightarrow_{\mathbf{S}} & \mathsf{X\backslash Z}{:}\lambda z.fz(gz) \\
\textit{Generalized Forward Composition:} & \mathsf{X/Y}{:}f & \mathsf{(Y/W)/Z}{:}g & \Rightarrow_{\mathbf{B}^2} & \mathsf{(X/W)/Z}{:}\lambda z.\lambda w.f(gzw) \\
\end{array} \tag{5}$$

For example, the following is the derivation of a relative clause related to (4):

$$\begin{array}{cccccl}
\textit{money} & \textit{that} & \textit{I} & \textit{give} & \textit{them} & \\
\hline
\mathsf{N} & \mathsf{(N\backslash N)/(S/NP)} & \mathsf{NP} & \mathsf{((S\backslash NP)/NP)/NP} & \mathsf{NP} & \\
: money' & : \lambda p\lambda q\lambda x.px \wedge qx & : I' & : \lambda x\lambda y\lambda z.give'yxz & : them' & \\
\cline{3-3}
 & & \mathsf{S/(S\backslash NP)} : \lambda f.fI' & & & >\mathbf{T} \\
\cline{3-4}
 & & \multicolumn{2}{c}{\mathsf{(S/NP)/NP} : \lambda x\lambda y.give'yxI'} & & >\mathbf{B}^2 \\
\cline{3-5}
 & & \multicolumn{3}{c}{\mathsf{S/NP} : \lambda y.give'y\,them'I'} & > \\
\cline{2-4}
 & \multicolumn{3}{c}{\mathsf{N\backslash N} : \lambda q\lambda x.give'x\,them'I' \wedge qx} & & > \\
\cline{1-5}
\multicolumn{5}{c}{\mathsf{N} : \lambda x.give'x\,them'I' \wedge money'x} & < \\
\end{array} \tag{6}$$

We will see further examples of their use later. Such rules induce additional derivational ambiguity, even in canonical sentences like (4). However, our translation

[4] Application of the two rules is indicated by underlines distinguished as < and >, respectively.

algorithm yields normal form derivations (Hepple and Morrill 1989; Wittenburg and Wall 1991; König 1994; Eisner 1996), which use composition and type-raising only when syntactically necessary. For coordination, we will use a binarized version of the following ternary rule schema:[5]

$$\textit{Coordination:} \qquad \mathsf{X}{:}\, f \quad \textit{conj} \quad \mathsf{X}{:}\, g \qquad \Rightarrow_{\&} \quad \mathsf{X}: \lambda x.fx \wedge gx \tag{7}$$

For further explanation and linguistics and computational motivation for this theory of grammar, the reader is directed to Steedman (1996, 2000).

2.3 Head-Dependency Structure in CCGbank

The syntactic derivations in CCGbank are accompanied with bilexical head-dependency structures, which are defined in terms of the lexical heads of functor categories and their arguments. The derivation in (6) corresponds to the following dependency structure, which includes the long-range dependency between *give* and *money*:

$$\begin{array}{l} \langle \textit{that}, \mathsf{(N\backslash N)/(S[dcl]/NP)}, 1, \textit{money}\rangle, \\ \langle \textit{that}, \mathsf{(N\backslash N)/(S/NP)}, 2, \textit{give}\rangle, \\ \langle \textit{give}, \mathsf{((S[dcl]\backslash NP_1)/NP_2)/NP_3}, 1, \textit{I}\rangle, \\ \langle \textit{give}, \mathsf{((S[dcl]\backslash NP_1)/NP_2)/NP_3}, 2, \textit{money}\rangle, \\ \langle \textit{give}, \mathsf{((S[dcl]\backslash NP_1)/NP_2)/NP_3}, 3, \textit{them}\rangle \end{array} \tag{8}$$

The dependency structures in CCGbank are intended to include all non-anaphoric local and long-range dependencies relevant to determining semantic predicate–argument relations, and hence approximate more fine-grained semantic representations. In this, they differ crucially from the bilexical surface dependencies used by the parsing models of Collins (1999) and Charniak (2000) and returned by the dependency parser of McDonald, Crammer, and Pereira (2005). In order to obtain such non-local dependencies, certain types of lexical category such as relative pronouns or raising and control verbs require additional coindexation information (described subsequently).

We believe that CCGbank's extensive annotation of non-local predicate–argument dependencies is one of its most useful features for researchers using other expressive grammar formalisms, including LFG, HPSG, and TAG, facilitating comparisons in terms of error analyses of particular constructions or types of dependency, such as non-subject extracted relative clauses. Because these dependency structures provide a suitable approximation of the underlying semantics, and because each interpretation unambiguously corresponds to one dependency structure (but may be obtained from multiple, equivalent, derivations), we furthermore follow Lin (1998) and Carroll, Minnen, and Briscoe (1999) in regarding them as a fairer, and ultimately more useful,

[5] In the semantics of this rule, $\wedge$ stands for the usual pointwise recursion over logical conjunction (Rooth and Partee 1982). Baldridge (2002), following Hepple (1990), Morrill (1994), and Moortgat (1997), advocates a variant of CCG where categories and rules are further specified by modalities that limit the applicability of rules such as composition. This variant allows a more elegant account in which conjunctions have categories of the form $(\mathsf{X}\backslash_{\star}\mathsf{X})/_{\star}\mathsf{X}$, in which the $\star$ modality prevents the rules in (5) from applying, because they are variously restricted by $\times$ ("crossed") and $\diamond$ ("harmonic") modalities. In the current version of CCGbank, such modalities are ignored, at some cost to generative soundness.

standard against which to evaluate the output of parsers trained on CCGbank than the syntactic derivations themselves.

3. The Penn Treebank

The Wall Street Journal subcorpus of the Penn Treebank contains about 50,000 sentences, or 1 million words, annotated with part-of-speech tags and phrase-structure trees:

```
(S (NP-SBJ (PRP He))                                              (9)
           (VP (VBZ is)
           (VP (ADVP (RB just))
               (VBG passing)
               (NP (DT the) (NN buck))
               (PP-DIR (TO to)
                   (NP (JJ young) (NNS people)))))
    (. .))
```

These trees are relatively flat: modals and auxiliaries introduce a new VP level, whereas verb modifiers and arguments typically appear all at the same level, as sisters of the main verb. A similarly flat annotation style is adopted at the sentence level. NPs are flat as well, with all complex modifiers appearing at the same NP level, and compound nouns typically lacking any internal structure.

The translation algorithm needs to identify syntactic heads, and has to distinguish between complements and modifiers. In the Treebank, this information is not explicit. Although some non-terminal nodes carry additional function tags, such as `-SBJ` (subject) or `-TMP` (temporal modifier), truly problematic cases such as prepositional phrases are often marked with tags such as `-CLR` ("closely related") or `-DIR` ("direction"), which are not always reliable or consistent indicators that a constituent is a modifier or an argument.

The Treebank uses various types of null elements and traces to encode non-local dependencies. These are essential for our algorithm since they make it possible to obtain correct CCG derivations for relative clauses, *wh*-questions, and coordinate constructions such as right node raising. Their treatment is discussed in Sections 6.2 and 6.3.

4. The Basic Translation Algorithm

In order to obtain CCG derivations from the Penn Treebank, we need to define a mapping from phrase structure trees to CCG derivations, including a treatment of the null elements in the Treebank. We also need to modify the Treebank where its syntactic analyses differ from CCG, and clean up certain sources of noise that would otherwise result in incorrect CCG derivations.

We will begin by ignoring null elements, and assume that Penn Treebank trees are entirely consistent with CCG analyses. The basic algorithm then consists of four steps:

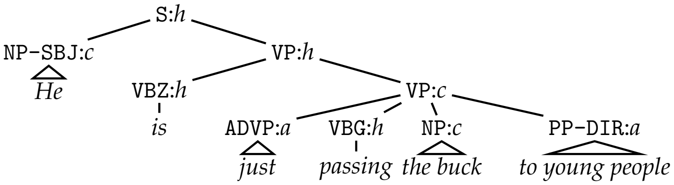
```
foreach tree $\tau$:                                              (10)
        determineConstituentTypes($\tau$);
        makeBinary($\tau$);
        assignCategories($\tau$);
        assignDependencies($\tau$);
```

Similar algorithms for phrase-structure trees without traces or other null elements have been suggested by Buszkowski and Penn (1990) and Osborne and Briscoe (1998).

We illustrate this basic algorithm using the previous example (9). Then we will extend this algorithm to deal with coordination, and introduce a modification to cope with the fact that certain word classes, such as participials, can act as modifiers of a large number of constituent types. Section 5 summarizes the most important preprocessing steps that were necessary to obtain the desired CCG analyses from the Treebank trees. Section 6 extends this basic algorithm to deal with the null elements in the Treebank.

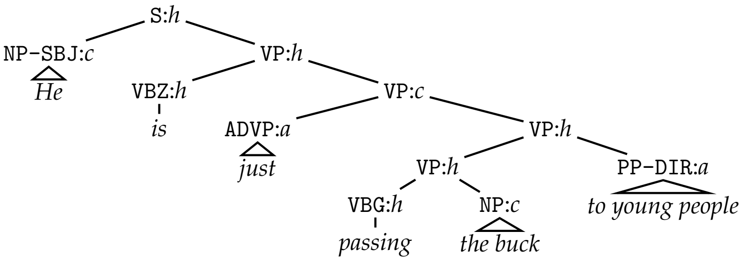
4.1 Determining Constituent Types: Heads, Complements, and Adjuncts

First, the constituent type of each node (head (*h*), complement (*c*), or adjunct (*a*)) is determined, using heuristics adapted from Magerman (1994) and Collins (1999), which take the label of a node and its parent into account.[6] We assume that `NP` daughters of `VPs` are complements, unless they carry a function tag such as `-LOC`, `-DIR`, `-TMP`, and so on, but treat all `PPs` as adjuncts unless they carry the `-CLR` function tag. In our example, we therefore treat *passing* as transitive, even though it should subcategorize for the PP:



4.2 Binarizing the Tree

Next, the tree is binarized:



This binarization process inserts dummy nodes into the tree such that all children to the left of the head branch off in a right-branching tree, and then all children to the right of the head branch off in a left-branching tree.[7]

[6] We had to modify these heuristics in minor ways. A complete list of our rules is in the CCGbank manual (Hockenmaier and Steedman 2005).

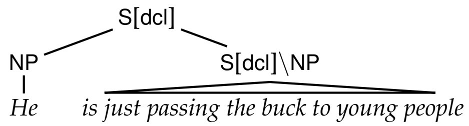
[7] In order to guarantee that sentence-final punctuation marks (omitted in the example for space reasons) modify the sentence rather than the VP, we first insert an additional `S` node that spans everything but the sentence-final punctuation mark.

4.3 Assigning Categories

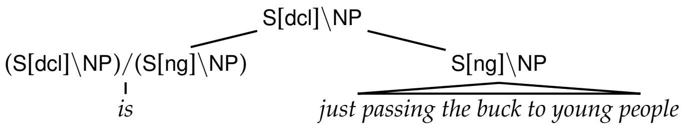
We assign CCG categories to the nodes in this binary tree in the following manner:

4.3.1 The Root Node. The category of the root node is determined by the label of the root of the Treebank tree (e.g., {VP} → S\NP, {S, SINV, SQ} → S).[8] If the root node has the category S, it typically carries a feature that distinguishes different types of sentences, such as declaratives (S[dcl]), *wh*-questions (S[wq]), yes–no questions (S[q]), or fragments (S[frg]). In our running example, the root is S[dcl], because its Treebank label is S, and its head word, the auxiliary, has the POS tag VBZ.

4.3.2 Head and Complement. The category of a complement child is defined by a similar mapping from Treebank labels to categories, for example, {NP} → NP, {PP} → PP.[9] The CCG category of the head is a function which takes the category of the complement as argument and returns the category of the parent node. The direction of the slash is given by the position of the complement relative to the head:

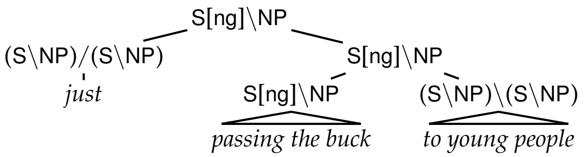


The VP that is headed by the main verb *passing* is a complement of the auxiliary. Because the POS tag of *passing* is VBG, the CCG category of the complement VP is S[ng]\NP (present participle) and the lexical category of *is* is therefore (S[dcl]\NP)/(S[ng]\NP):



Other VP features include [to] (*to* infinitival), [b] (bare infinitival), S[pt] (past participle), [pss] (passive), or [ng] (present participle).

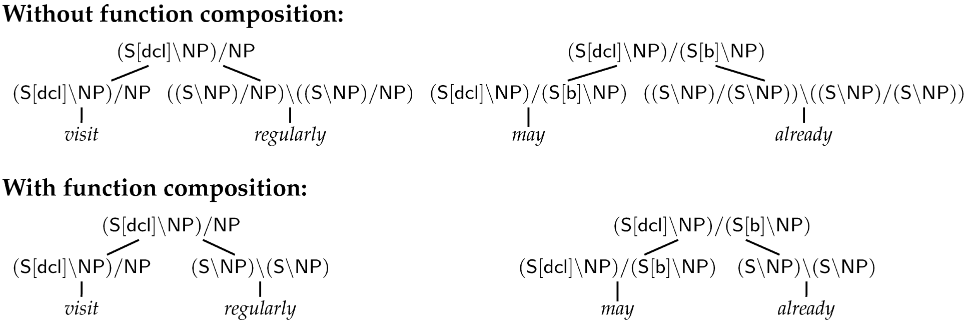
4.3.3 Head and Adjunct. According to the Treebank annotation and the assumptions of the algorithm, our example has two VP adjuncts: the adverb *just*, and, because of its -DIR function tag, the PP *to young people*. In both cases, the adjunct category depends on the category of the parent, and the category of the head child is copied from the parent:



Given a parent category C, the category of an adjunct child is a unary functor C′/C′ if the adjunct child is to the left of the head child (a premodifier), or C′\C′ if it is to the right

[8] Every Treebank tree is in fact rooted in an unlabeled node. This node is kept in the translation and assigned the label TOP.

[9] Generally, the Treebank label determines the category of complements. Function tags, such as -SBJ, are currently not reflected in the categories. However, we use *EXP* null elements and the EX POS-tag to assign categories NP[expl] and NP[thr] to expletive *it* or *there*.



Without function composition:

With function composition:

Figure 1
Function composition reduces the number of lexical categories of adjuncts.

of the head (a postmodifier). In most cases, the category C′ is equal to the parent category C without any features such as [dcl], [ng], and so forth, and the modifier combines with the head via simple function application. As shown in Figure 1, in many cases, a more elegant (and general) analysis can be obtained if we allow modifiers to compose with the head. For example, *regularly* has the category (S\NP)\(S\NP) in sentences such as *I visit certain places regularly*, because it modifies the verb phrase *visit certain places*, which has the category S[dcl]\NP. But in the corresponding relative clause *places that I visit regularly* or with heavy NP shift (*I visit regularly certain places in Europe*), *regularly* modifies *visit*, that is, a constituent with category (S[dcl]\NP)/NP. Without function composition, the category of *regularly* would have to be ((S\NP)/NP)\((S\NP)/NP), but (crossed) composition allows the ordinary category (S\NP)\(S\NP) to also work in this case.

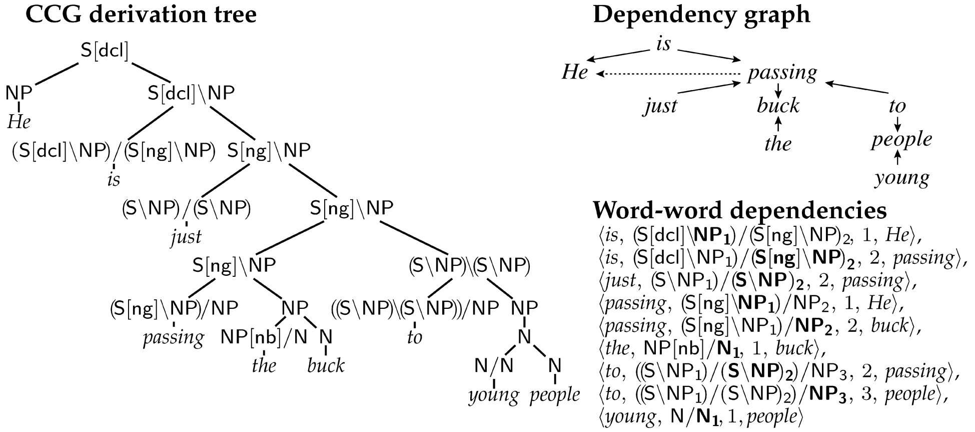
Therefore, if the parent (and head) category C is of the form X/$, the algorithm strips off all outermost forward arguments /$ (and syntactic features) from C to obtain C′. Similarly, if C is of the form X\$, all outermost backward arguments \$ (and syntactic features) are stripped off from C to obtain C′.

4.3.4 Head and Punctuation Mark. With the exception of some dashes and parentheses (see Section 4), the category of a punctuation mark is identical to its POS tag, and the head has the same category as its parent.

4.3.5 The Final Derivation. Figure 2 shows the complete CCG derivation of our example. The category assignment procedure corresponds to a top-down normal-form derivation, which almost always uses function application. In the basic case presented here, composition is only used to provide a uniform analysis of adjuncts. Long-range dependencies represented in the Penn Treebank by traces such as *T* and *RNR* require extensions to the basic algorithm, which result in derivations that make use of type-raising, composition, and (occasionally) substitution rules like those in (5) wherever syntactically necessary. We defer explanation of these rules until Section 6, which presents the constructions that motivate them.

4.4 Assigning the Dependency Structure

Finally, we need to obtain the word–word dependencies which approximate the underlying predicate–argument structure. This is done by a bottom-up procedure, which simply retraces the steps in the CCG derivation that we have now obtained.



Word-word dependencies

$\langle$*is*, $(\mathsf{S[dcl]\backslash \mathbf{NP_1}})/(\mathsf{S[ng]\backslash NP})_2$, 1, *He*$\rangle$,
$\langle$*is*, $(\mathsf{S[dcl]\backslash NP_1})/(\mathbf{S[ng]\backslash NP})_{\mathbf{2}}$, 2, *passing*$\rangle$,
$\langle$*just*, $(\mathsf{S\backslash NP_1})/(\mathbf{S\backslash NP})_{\mathbf{2}}$, 2, *passing*$\rangle$,
$\langle$*passing*, $(\mathsf{S[ng]\backslash \mathbf{NP_1}})/\mathsf{NP_2}$, 1, *He*$\rangle$,
$\langle$*passing*, $(\mathsf{S[ng]\backslash NP_1})/\mathbf{NP_2}$, 2, *buck*$\rangle$,
$\langle$*the*, $\mathsf{NP[nb]}/\mathbf{N_1}$, 1, *buck*$\rangle$,
$\langle$*to*, $((\mathsf{S\backslash NP_1})/(\mathbf{S\backslash NP})_{\mathbf{2}})/\mathsf{NP_3}$, 2, *passing*$\rangle$,
$\langle$*to*, $((\mathsf{S\backslash NP_1})/(\mathsf{S\backslash NP})_2)/\mathbf{NP_3}$, 3, *people*$\rangle$,
$\langle$*young*, $\mathsf{N}/\mathbf{N_1}$, 1, *people*$\rangle$

Figure 2
The CCG derivation with corresponding dependencies and dependency graph for example (9).

All categories in CCGbank, including results and arguments of complex categories, are associated with a corresponding list of lexical heads. This list can be empty (in the case of yet uninstantiated arguments of functor categories), or it can consist of one or more tokens. Lexical categories have one lexical head, the word itself—for example, *He* for the first NP, and *is* for the (S[dcl]\NP)/(S[b]\NP). All dependencies are defined in terms of the heads of lexical functor categories and of their arguments. In order to distinguish the slots filled by different arguments, we number the arguments of complex lexical categories from left to right in the category notation (that is, from innermost to outermost argument in a purely applicative derivation), for example, $(\mathsf{S[ng]\backslash NP_1})/\mathsf{NP_2}$, or $((\mathsf{S[b]\backslash NP_1})/(\mathsf{S[to]\backslash NP})_2)/\mathsf{NP_3}$.

In lexical functor categories such as that of the auxiliary, (S[dcl]\NP)/(S[b]\NP), the lexical head of all result categories (S[dcl]\NP and S[dcl]) is identical to the lexical head of the entire category (i.e., *is*). But in functor categories that represent modifiers, such as the adverb (S\NP)/(S\NP), the head of the result (the modified verb phrase) comes from the argument (the unmodified verb phrase). We use indices on the categories to represent this information: $(\mathsf{S\backslash NP})_i/(\mathsf{S\backslash NP})_i$. In CCGbank, modifier categories are easily identified by the fact that they are of the form X|X or (X|X)|... (with | either / or \), where X does not have any of the features described previously, such as [dcl], [b]. Similarly, determiners (*the*) take a noun (N, *buck*) as argument to form a (non-bare) noun phrase whose lexical head comes from the noun: $\mathsf{NP[nb]}_i/\mathsf{N}_i$. Thus, the lexical head of the noun phrase *the buck* is *buck*, not *the*.

We also use this coindexation mechanism for lexical categories that project nonlocal dependencies. For instance, the category of the auxiliary, (S[dcl]\NP)/(S[ng]\NP), mediates a dependency between the subject (*He*) and the main verb (*passing*). Like all lexical categories of auxiliaries, modals and subject-raising verbs, the head of the subject NP is coindexed with the head of subject inside the VP argument: $(\mathsf{S[dcl]\backslash NP_i})/(\mathsf{S[ng]\backslash NP_i})$. The set of categories that project such dependencies is not acquired automatically, but is given (as a list of category templates) to the algorithm which creates the actual dependency structures. A complete list of the lexical entries in sections 02–21 of the Treebank which use this coindexation mechanism to project nonlocal dependencies is given in the CCGbank manual (Hockenmaier and Steedman 2005). We believe that in practice this mechanism is largely correct, even though it is based on the (fundamentally flawed) assumption that all lexical categories that have the same syntactic type project the same dependencies. It may be possible to use the indices on the PRO-null elements (*-1) in the Treebank to identify and resolve ambiguous cases; we leave this to future research.[10]

Function application and composition typically result in the instantiation of the lexical head of an argument of some functor category, and therefore create new dependencies, whereas coordination creates a new category whose lexical head lists are concatenations of the head lists of the conjuncts.

When the $(\mathsf{S[ng]\backslash NP_1})/\mathsf{NP_2}$ *passing* is combined with the NP *the buck*, the lexical head of the $\mathsf{NP_2}$ is instantiated with *buck*. Similarly, when the adverb *just* $(\mathsf{S\backslash NP_1})/(\mathsf{S\backslash NP})_2$ is applied to *passing the buck*, a dependency between *just* and *passing* is created:

$$\langle \textit{passing}, (\mathsf{S[ng]\backslash NP})/\mathsf{NP}, 2, \textit{buck}\rangle,\ \langle \textit{just}, (\mathsf{S\backslash NP})/(\mathsf{S\backslash NP}), 2, \textit{passing}\rangle \tag{11}$$

However, because $(\mathsf{S\backslash NP_1})/(\mathsf{S\backslash NP})_2$ is a modifier category, the head of the resulting $\mathsf{S[ng]\backslash NP}$ is *passing*, not *just* (and no dependency is established between *just* and its $\mathsf{NP_1}$). In the next step, this $\mathsf{S[ng]\backslash NP}$ is combined with the auxiliary $(\mathsf{S[dcl]\backslash NP_1})/(\mathsf{S[ng]\backslash NP})_2$. The NP in the $(\mathsf{S[ng]\backslash NP})_2$ argument of the auxiliary unifies with the (uninstantiated) $\mathsf{NP_1}$ argument of *passing*. Because the NP in the $(\mathsf{S[ng]\backslash NP})_2$ is also coindexed with the subject $\mathsf{NP_1}$ of the auxiliary, the NP of the resulting $\mathsf{S[dcl]\backslash NP}$ now has two unfilled dependencies to the subject $\mathsf{NP_1}$ of *is* and *passing*. When the entire verb phrase is combined with the subject, *He* fills both slots:

$$\begin{array}{l} \langle \textit{passing}, (\mathsf{S[ng]\backslash NP})/\mathsf{NP}, 1, \textit{He}\rangle \\ \langle \textit{is}, (\mathsf{S[dcl]\backslash NP})/(\mathsf{S[ng]\backslash NP}, 1, \textit{He}\rangle \end{array} \tag{12}$$

Figure 2 shows the resulting CCG derivation and the corresponding list of word–word dependencies for our example sentence. It is the latter structure that we claim approximates for present purposes the predicate–argument structure or interpretation of the sentence, and provides the gold standard against which parsers can be evaluated.

4.5 Coordination

In order to deal with coordination, both the tree binarization and the category assignment have to be modified.

In CCGbank, coordination is represented by the following binary rule schemata, rather than the ternary rule (7)—compare to Steedman (1989):[11]

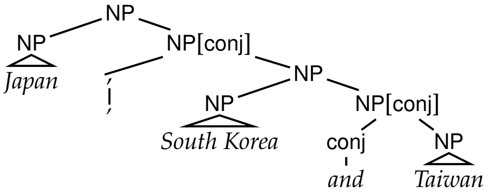
$$\begin{array}{l} \text{a. } \mathsf{conj}\quad \mathsf{X} \Rightarrow \mathsf{X[conj]} \\ \text{b. } ,\quad \mathsf{X} \Rightarrow \mathsf{X[conj]} \\ \text{c. } \mathsf{X}\quad \mathsf{X[conj]} \Rightarrow \mathsf{X} \end{array} \tag{13}$$

In order to obtain this analysis from Treebank trees, a separate node that spans only the conjuncts and the conjunction or punctuation marks (comma, semicolon) is inserted if necessary. Identifying the conjuncts often requires a considerable amount of preprocessing. These trees are then transformed into strictly right-branching binary trees. The

[10] That our assumption that coindexation can be determined deterministically from the syntactic types alone is incorrect is most obvious in the case of control verbs: Both *promise* and *persuade* have the syntactic category $((\mathsf{S\backslash NP})/(\mathsf{S[to]\backslash NP}))/\mathsf{NP}$, yet for *persuade*, the subject of the *to*-VP should be coindexed with the object NP, whereas for *promise*, it should be coindexed with the subject. However, all control verbs that we identified in CCGbank are object control.

[11] As noted earlier, in Baldridge's (2002) modal version of CCG, conjunctions have categories of the form $(\mathsf{X\backslash X})/\mathsf{X}$, but without modalities this category leads to overgeneralization.

dummy nodes inserted during binarization receive the same category as the conjuncts, but additionally carry a feature [conj]:



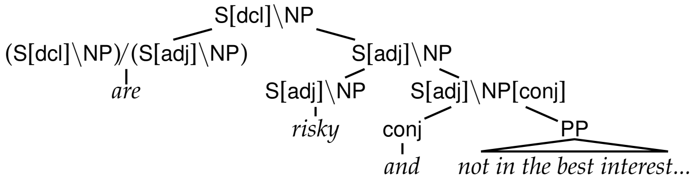
An additional modification of the grammar is necessary to deal with "unlike coordinate phrases" (UCP), namely, coordinate constructions where the conjuncts do not belong to the same syntactic category:

```
(VP (VBP are)                                                         (14)
    (UCP-PRD (ADJP risky)
             (CC and)
             (PP not in the best interest of the investing public)))
```

Such constructions are difficult for any formalism. This phenomenon could be handled elegantly with a feature hierarchy over categories as proposed by Copestake (2002), Villavicencio (2002), and McConville (2007). Because the induction of such a hierarchy was beyond the scope of our project, we modify our grammar slightly, and allow the algorithm to use instantiations of a special coordination rule schema, such as:

$$\mathsf{conj}\quad \mathsf{PP} \Rightarrow \mathsf{S[adj]\backslash NP[conj]} \tag{15}$$

This enables us to analyze the previous example as:



4.6 Type-Changing Rules for Clausal Adjuncts

In CCG, all language-specific information is associated with the lexical categories of words. There are many syntactic regularities associated with word classes, however, which may potentially generate a large number of lexical entries for each item in that class. One particularly frequent example of this is clausal adjuncts.

Figure 3 illustrates how the basic algorithm described above leads to a proliferation of adjunct categories. For example, a past participle such as *used* would receive a different category in a reduced relative like Figure 3(a) from its standard category (S[pss]\NP)/(S[to]\NP). As a consequence, modifiers of *used* would also receive different categories depending on what occurrence of *used* they modify. This is undesirable, because we are only guaranteed to acquire a complete lexicon if we have seen all participles (and their possible modifiers) in all their possible surface positions. Similar regularities have been recognized and given a categorial analysis by Carpenter (1992), who advocates lexical rules to account for the use of predicatives as adjuncts. In a statistical model, the parameters for such lexical rules are difficult to estimate. We therefore follow the approach of Aone and Wittenburg (1990) and implement these type-changing

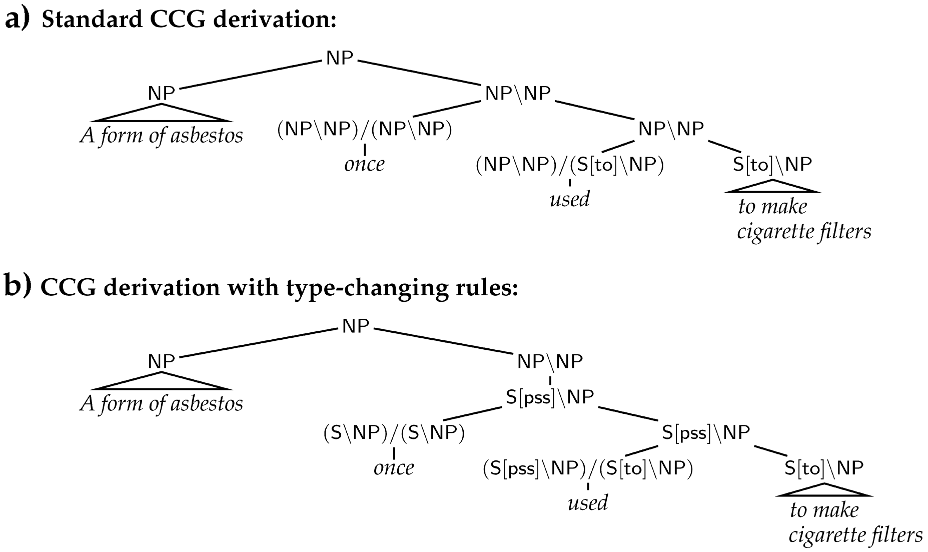


Figure 3
Type-changing rules reduce the number of lexical category types required for complex adjuncts.

operations in the derivational syntax, where these generalizations are captured in a few rules. If these rules apply recursively to their own output, they can generate an infinite set of category types, leading to a shift in generative power from context-free to recursively enumerable (Carpenter 1991, 1992). Like Aone and Wittenburg, we therefore consider only a finite number of instantiations of these type-changing rules, namely those which arise when we extend the category assignment procedure in the following way: For any sentential or verb phrase modifier (an adjunct with label S or SBAR with null complementizer, or VP) to which the original algorithm assigns category X|X, apply the following type-changing rule (given in bottom-up notation) in reverse:

$$\mathrm{S\$} \Rightarrow \mathrm{X|X} \qquad (16)$$

where S$ is the category that this constituent obtains if it is treated like a head node by the basic algorithm. S$ has the appropriate verbal features, and can be S\NP or S/NP. Some of the most common type-changing rules are the following, for various types of reduced relative modifier:

a. $\mathrm{S[pss]\backslash NP_i} \Rightarrow \mathrm{NP_i\backslash NP_i}$ (17)
"workers [*exposed to it*]*"*

b. $\mathrm{S[adj]\backslash NP_i} \Rightarrow \mathrm{NP_i\backslash NP_i}$
"a forum [*likely to bring attention to the problem*]*"*

c. $\mathrm{S[ng]\backslash NP_i} \Rightarrow \mathrm{NP_i\backslash NP_i}$
"signboards [*advertising imported cigarettes*]*"*

d. $\mathrm{S[ng]\backslash NP_i} \Rightarrow \mathrm{(S\backslash NP_i)\backslash(S\backslash NP_i)}$
"become chairman, [*succeeding Ian Butler*]*"*

e. $\mathrm{S[dcl]/NP_i} \Rightarrow \mathrm{NP_i\backslash NP_i}$
"the millions of dollars [*it generates*]*"*

In order to obtain the correct predicate–argument structure, the heads of corresponding arguments in the input and output category are unified (as indicated by coindexation).

In written English, certain types of NP-extraposition require a comma before or after the extraposed noun phrase:

*Factories booked $236.74 billion in orders in September, [*$_{\mathrm{NP}}$ *nearly the same as the $236.79 billion in August]* (18)

Because any predicative noun phrase could be used in this manner, this construction is also potentially problematic for the coverage of our grammar and lexicon. However, the fact that a comma is required allows us to use a small number of binary type-changing rules (which do not project any dependencies), such as:

$$\begin{array}{lllll} \mathsf{NP} & , & \Rightarrow & \mathsf{S/S} \\ , & \mathsf{NP} & \Rightarrow & \mathsf{S\backslash S} \\ , & \mathsf{NP} & \Rightarrow & \mathsf{(S\backslash NP)\backslash(S\backslash NP)} \end{array}$$

5. Necessary Preprocessing Steps

The translation algorithm presumes that the trees in the Penn Treebank map directly to the desired CCG derivations. However, this is not always the case, either because of noise in the Treebank annotation, differences in linguistic analysis, or because CCG, like any other expressive linguistic formalism, requires information that is not present in the Treebank analysis. Before translation, a number of preprocessing steps are therefore required. Disregarding the most common preprocessing step (the insertion of a noun level, which is required in virtually all sentences), preprocessing affects almost 43% of all sentences. Here we summarize the most important preprocessing steps for those constructions that do not involve non-local dependencies. Preprocessing steps required for constructions involving non-local dependencies (i.e., traces or null elements in the Treebank) are mentioned in Section 6. Remaining problems are discussed in Section 7. More detailed and complete descriptions can be found in the CCGbank manual.

5.1 Dealing with Noise in the Treebank

Annotation errors and inconsistencies in the Treebank affect the quality of any extracted grammar or lexicon. This is especially true for formalisms with an extended domain of locality, such as TAG or CCG, where a single elementary tree or lexical category may contain information that is distributed over a number of distinct phrase-structure rules.

Part-of-Speech Tagging Errors. Ratnaparkhi (1996) estimates a POS tagging error rate of 3% in the Treebank. The translation algorithm is sensitive to these errors and inconsistencies, because POS tagging errors can lead to incorrect categories or to incorrect features on verbal categories (e.g., when a past participle is wrongly tagged as past tense). For instance, if a simple past tense form occurs in a verb phrase which itself is the daughter of a verb phrase whose head is an inflected verb, it is highly likely that it should be a past participle instead. Using the verb form itself and the surrounding context, we have attempted to correct such errors automatically. In 7% of all sentences, our algorithm modifies at least one POS tag.

Quotation Marks. Although not strictly coming under the heading of noise, quotation marks cause a number of problems for the translation algorithm. Although it is tempting to analyze them similarly to parentheticals, quotations often span sentence boundaries, and consequently quotation marks appear to be unbalanced at the sentence level. We therefore decided to eliminate them during the preprocessing stage.

5.2 Adding Structure to the Treebank Analyses

Unlike a hand-written grammar, the grammar that is implicit in a treebank has to cover all constructions that occur in the corpus. Expressive formalisms such as CCG provide explicit analyses that contain detailed linguistic information. For example, CCG derivations assign a lexical head to every constituent and define explicit functor–argument relations between constituents. In a phrase-structure grammar, analyses can be much coarser, and may omit more fine-grained structures if they are assumed to be implicit in the given analysis. Furthermore, constructions that are difficult to analyze do not need to be given a detailed analysis. In both cases, the missing information has to be added before a Treebank tree can be translated into CCG. If the missing structure is implicit in the Treebank analysis, this step is relatively straightforward, but constructions such as parentheticals, multiword expressions, and fragments require careful reanalysis in order to avoid lexical coverage problems and overgeneration.

Detecting Coordination. Although the Treebank does not explicitly indicate coordination, it can generally be inferred from the presence of a conjunction. However, in list-like nominal coordinations, the conjuncts are only separated by commas or semicolons, and may be difficult to distinguish from appositives. There are also a number of verb-phrase or sentential coordinations in the Treebank where shared arguments or modifiers simply appear at the same level as conjuncts and the conjunction:[12]

```
(VP (VBP meet)                                        (19)
    (CC or)
    (VBP exceed)
    (NP their 1989 spending))
```

In CCG, the conjuncts and conjunction form a separate constituent. In 1.8% of all sentences, additional preprocessing is necessary to obtain this structure.

Noun Phrases and Quantifier Phrases. In the Penn Treebank, non-recursive noun phrases have remarkably little internal structure:

```
(NP (DT the) (NNP Dutch) (VBG publishing) (NN group))          (20)
```

Some, but not all, of the structure that is required to obtain a linguistically adequate analysis can be inferred (semi-)automatically. The CCGbank grammar distinguishes noun phrases, NP, from nouns, N, and treats determiners (*the*) as functions from nouns

[12] Other examples include adverbial expressions such as *therefore, so, even* that appear between the conjunction and the second conjunct.

to noun phrases (NP[nb]/N). Therefore, we need to insert an additional noun level, which also includes the adjuncts *Dutch* and *publishing*, which receive both the category N/N:

```
(NP (DT the)                                              (21)
    (NOUN (NNP Dutch) (VBG publishing) (NN group)))
```

However, because nominal compounds in the Treebank have no internal bracketing, we always assume a right-branching analysis, and are therefore not able to obtain the correct dependencies for cases such as *(lung cancer) deaths*.

QPs ("quantifier phrases") are another type of constituent where the Treebank annotation lacks internal structure:

```
(QP (IN between) (CD 3) (NN %) (CC and) (CD 5) (NN %))    (22)
```

We use a number of heuristics to identify the internal structure of these constituents—for example, to detect conjuncts and prepositions. The above example is then re-bracketed:

```
(QP (IN between)                                          (23)
    (QP (QP (CD 3) (NN %))
            (CC and)
            (QP (CD 5) (NN %)))
```

Fragments. 1.24% of the sentences in the Penn Treebank correspond to or contain fragmentary utterances (labeled FRAG), for which no proper analysis could be given:

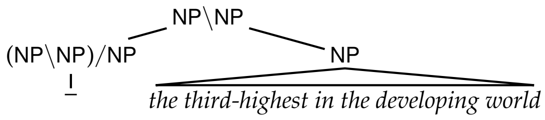
```
a. (FRAG (NP The next province) (. ?))                    (24)
b. (SBARQ (WRB how) (RP about) (FRAG (NP the New Guinea Fund)) (. ?)
```

FRAGs are often difficult to analyze, and the annotation is not very consistent. The CCGbank manual lists heuristics that we used to infer additional structure. For example, if a node is labeled FRAG, and there is only one daughter (and potentially an end-of-sentence punctuation mark), as in the first example, we treat the tree as if it was labeled with the label of its daughter (NP in this case).

Parentheticals. Parentheticals are insertions that are often enclosed in parentheses, or preceded by a dash. Unless the parenthetical element itself is of a type that could be a modifier by itself (e.g., a PP), we assume that the opening parenthesis or first dash takes the parenthetical element as argument and yields a modifier of the appropriate type:

```
(PRN (: --)                                               (25)
     (NP (NP the third-highest)
         (PP-LOC in the developing world)))
```

This results in the following derivation, which ignores the fact that parentheses are usually balanced (Nunberg 1990):



We use a similar treatment for other constituents that appear after colons and dashes, such as sentence-final appositives, or parentheticals that are not marked as PRN. Overall, these changes affect 8.7% of all sentences.

Multi-Word Expressions. Under the assumption that every constituent has a lexical head that corresponds to an individual orthographic word, multi-word expressions require an analysis where one of the items subcategorizes for a specific syntactic type that can only correspond to the other lexical item. We only attempted an analysis for expressions that are either very frequent or where the multi-word expression has a different subcategorization behavior from the head word of the expression. This includes some closed-class items (described in the CCGbank manual), including connectives (e.g., *as if, as though, because of*), comparatives (*so ADJ that, too ADJ to, at least/most/ . . . X*), monetary expressions, and dates, affecting 23.8% of all sentences.

5.3 Changing the Treebank Analyses

Additionally, there are a number of constructions whose Treebank annotation differs from the standard CCG analysis for linguistic reasons. This includes small clauses, as well as pied-piping, subject extraction from embedded sentences and argument cluster coordination (discussed in Section 6).

Small Clauses. The Treebank treats constructions such as the following as small clauses:

```
(S (NP-SBJ that volume)                                    (26)
 (VP (VBZ makes)
     (S (NP-SBJ it)
        (NP-PRD the largest supplier...in Europe))))
```

Pollard and Sag (1992) and Steedman (1996) argue against this analysis on the basis of extractions like *what does the country want forgiven*, which suggest that these cases should rather be treated as involving two complements. We eliminate the small clause, and transform the trees such that the verb takes both NP children of the small clause as complements, thereby obtaining the lexical category ((S[dcl]\NP)/NP)/NP for *makes*. Because our current grammar treats predicative NPs like ordinary NPs, we are not able to express the relationship between *it* and *supplier*, or between *pool* and *hostage*. A correct analysis would assign a functor category S[nom]\NP (or perhaps NP[prd]\NP) to predicative NP arguments of verbs like *makes*, not only in these examples, but also in copular sentences and appositives. The other case where small clauses are used in the Treebank includes absolute *with* and *though* constructions (*with the limit in effect*). Here, we also assume that the subordinating conjunction takes the individual constituents in the small clause as complements, and *with* obtains therefore the category ((S/S)/PP)/NP. Again, a predicative analysis of the PP might be desirable in order to express the dependencies between *limit* and *in effect*. Eliminating small clauses affects 8.2% of sentences.

6. Long-Range Dependencies in the Treebank

The treatment of non-local dependencies is one of the most important points of difference between grammar formalisms. The Treebank uses a large inventory of null element types and traces, including coindexation to represent long-range dependencies.

Because standard Treebank parsers use probabilistic versions of context-free grammar, they are generally trained and tested on a version of the Treebank in which these null elements and indices are deleted or ignored, or, in the case of Collin's (1999) Model 3, only partially captured. Non-local dependencies are therefore difficult to recover from their output. In CCG, long-range dependencies are represented without null elements or traces, and coindexation is restricted to arguments of the same lexical functor category. Although this mechanism is less expressive than the potentially unrestricted coindexation used in the Treebank, it allows parsers to recover non-anaphoric long-range dependencies directly, without the need for further postprocessing or trace insertion.

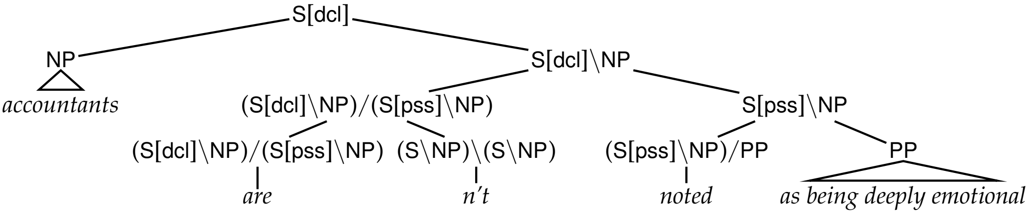
6.1 Passive, Control, Raising and Extraposition

Passive. In the Treebank, the surface subject of a passive sentence is coindexed with a $*$ null element in direct object position:

```
(S (NP-SBJ-1 accountants)                                    (27)
   (VP (VBP are)
   (RB n't)
   (VP (VBN noted)
       (NP-2 (-NONE- *-1))
       (PP-CLR as being deeply emotional))))
```

Our translation algorithm uses the presence of the $*$ null element to identify passive mode, but ignores it otherwise, assigning the CCG category S[pss]\NP to *noted*.[13] The dependency between the subject and the participial is mediated through the lexical category of the copula, (S[dcl]\NP$_i$)/(S[pss]\NP$_i$) (with the standard semantics $\lambda p \lambda x.px$).[14] In order to reduce lexical ambiguity and deal with data sparseness, we treat optional *by*-PPs which contain the "logical" subject (NP-LGS) as adjuncts rather than arguments of the passive participle.[15]

Here is the resulting CCG derivation, together with its dependency structure:



⟨*are*, (S[dcl]\NP)/(S[pss]\NP), 1, *accountants*⟩, ⟨*are*, (S[dcl]\NP)/(S[pss]\NP), 2, *noted*⟩
⟨*n't*, (S\NP)\(S\NP), 2, *are*⟩, ⟨*noted*, (S[pss]\NP)/PP, 1, *accountants*⟩, ⟨*noted*, (S[pss]\NP)/PP, 2, *as*⟩

[13] In the case of verbs like *pay for*, which take a PP argument, the null element appears within the PP. In order to obtain the correct lexical category of *paid*, (S[pss]\NP)/(PP/NP), we treat the null element like an argument of the preposition and percolate it up to the PP level.

[14] We assume that the fact that the subject NP argument of passive participials with category S[pss]\NP identifies the patient, rather than agent, is represented in the semantic interpretation of *noted*, for example, $\lambda x.noted'x\ one'$, where *one′* is simply a placeholder for a bindable argument, like the relational grammarians' *chômeur* relation.

[15] Extractions such as *Who was he paid by* require the *by*-PP to be treated as an argument, and it would in fact be better to use a lexical rule to generate (S[pss]\NP)/PP[by] from S[pss]\NP and vice versa.

Infinitival and Participial VPs, Gerunds. In the Treebank, participial phrases, gerunds, imperatives, and to-VP arguments are annotated as sentences with a $*$ null subject:

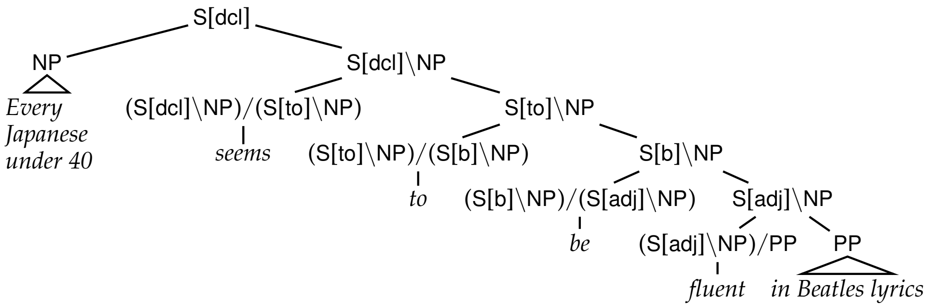
```
(PP (IN over)                                    (28)
    (S-NOM (NP-SBJ (-NONE- *))
           (VP (VBG cutting)
             (NP capital-gains taxes))))
```

We treat these like verb phrases (S\NP) with the appropriate feature ([b], [to], [ng], or [pt]), depending on the part-of-speech tag of the verb.

Control and Raising. CCGbank does not distinguish between control and raising. In the Treebank, subject-control and subject-raising verbs (e.g., *want* and *seem*) also take an S complement with a null subject that is coindexed with the subject of the main clause:

```
(S (NP-SBJ-1 Every Japanese under 40)                (29)
   (VP (VBZ seems)
      (S (NP-SBJ (-NONE- *-1))
         (VP to be fluent in Beatles lyrics))))
```

Because an S with an empty subject NP has category S\NP, we obtain the correct syntactic category $(\mathrm{S[dcl]}\backslash\mathrm{NP_i})/(\mathrm{S[to]}\backslash\mathrm{NP_i})$ for *seems*:



We ignore the coindexation in the Treebank, and treat all control verbs as non-arbitrary control. As indicated by the index *i*, we assume that all verbs which subcategorize for a verb phrase complement and take no direct object mediate a dependency between their subject and their complement. Because the copula and *to* mediate similar dependencies between their subjects and complements, but do not fill their own subject dependencies, *Japanese* has the following dependencies:

$$\begin{array}{l}\langle \textit{seems}, (\mathrm{S[dcl]}\backslash\mathrm{NP})/(\mathrm{S[to]}\backslash\mathrm{NP}), 1, \textit{Japanese}\rangle,\\ \langle \textit{fluent}, (\mathrm{S[adj]}\backslash\mathrm{NP})/\mathrm{PP}, 1, \textit{Japanese}\rangle\end{array} \tag{30}$$

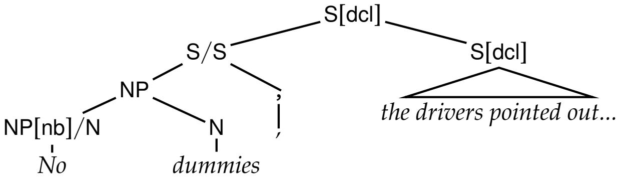
In the Treebank, object-raising verbs (*wants half the debt forgiven*) take a small clause argument with non-empty subject. Following our treatment of small clauses (see Section 5.3) we modify this tree so that we obtain the lexical category $(((\mathrm{S[dcl]}\backslash\mathrm{NP})/(\mathrm{S[pss]}\backslash\mathrm{NP_i}))/\mathrm{NP_i})$ for *wanted*, which mediates the dependency between *debt* and *forgiven*.[16]

[16] The English lexicon also has a very small number of subject control verbs like *promise*, bearing the category $((\mathrm{S[dcl]}\backslash\mathrm{NP_i})/(\mathrm{S[to]}\backslash\mathrm{NP}))/\mathrm{NP_i}$, which have to be treated specially. However, we did not find any such subject control verbs in the Wall Street Journal corpus.

Extraposition of Appositives. Appositive noun phrases can be extraposed out of a sentence or verb phrase, resulting in an anaphoric dependency. The Penn Treebank analyzes these as adverbial small clauses with a coindexed null subject:

```
(S (S-ADV (NP-SBJ *-1) (NP-PRD No dummies))                              (31)
   (, ,)
   (NP-SBJ-1 the drivers)
   (VP pointed out they still had space ...)
```

We also treat these appositives as sentential modifiers. However, the corresponding CCG derivation deliberately omits the dependency between *dummies* and *drivers*:[17]



This derivation uses one of the special binary type-changing rules (see Section 4.6) that takes into account that these appositives can only occur adjacent to commas.

6.2 Long-Range Dependencies Through Extraction

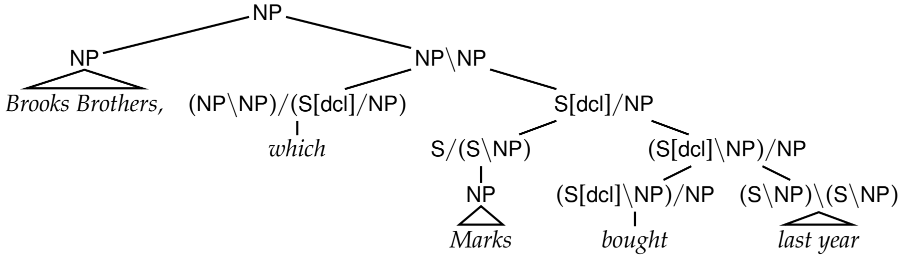
The Penn Treebank analyzes *wh*-questions, relative clauses, topicalization of complements, *tough* movement, cleft, and parasitic gaps in terms of movement. These constructions are frequent: The entire Treebank contains 16,056 `*T*` traces, including 8,877 `NP` traces, 4,120 `S` traces, 2,465 `ADVP` traces, 422 `PP` traces, and 210 other `*T*` traces. Sections 02–21 (39,604 sentences) contain 5,288 full subject relative clauses, as well as 459 full and 873 reduced object relative clauses. The dependencies involved in these constructions, however, are difficult to obtain from the output of standard parsers such as Collins (1999) or Charniak (2000), and require additional postprocessing that may introduce further noise and errors. In those cases where the trace corresponds to a "moved" argument, the corresponding long-range dependencies can be recovered directly from the correct CCG derivation.

In the Treebank, the "moved" constituent is coindexed with a trace (`*T*`), which is inserted at the extraction site:

```
(NP-SBJ (NP Brooks Brothers))                                          (32)
        (, ,)
        (SBAR (WHNP-1 (WDT which))
              (S (NP-SBJ NNP Marks))
                 (VP (VBD bought)
                     (NP (-NONE- *T*-1))
                     (NP-TMP last year))))))
```

[17] We regard this type of dependency as anaphoric rather than syntactic, on the basis of its immunity to such syntactic restrictions as subject islands.

CCG has a similarly uniform analysis of these constructions, albeit one that does not require syntactic movement. In the CCG derivation of the example, the relative pronoun has the category $(\mathsf{NP}_i\backslash\mathsf{NP}_i)/(\mathsf{S[dcl]}/\mathsf{NP}_i)$ whereas the verb *bought* just bears the standard transitive category (S[dcl]\NP)/NP. The subject NP and the incomplete VP combine via type-raising and forward composition into an S[dcl]/NP, which the relative pronoun then takes as its argument:



The coindexation on the lexical category of the relative pronoun guarantees that the missing object unifies with the modified NP, and we obtain the desired dependencies:

$$\begin{aligned}&\langle \textit{which}, \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, 1, \textit{Brothers}\rangle, \langle \textit{which}, \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, 1, \textit{bought}\rangle,\\ &\langle \textit{bought}, \mathsf{(S[dcl]\backslash NP)/NP}, 1, \textit{Marks}\rangle, \langle \textit{bought}, \mathsf{(S[dcl]\backslash NP)/NP}, 2, \textit{Brothers}\rangle\end{aligned} \tag{33}$$

This analysis of movement in terms of functors over incomplete constituents allows CCG to use the same category for the verb when its arguments are extracted as when they are in situ. This includes not only relative clauses and *wh*-questions, but also pied-piping, *tough* movement, topicalization, and clefts.

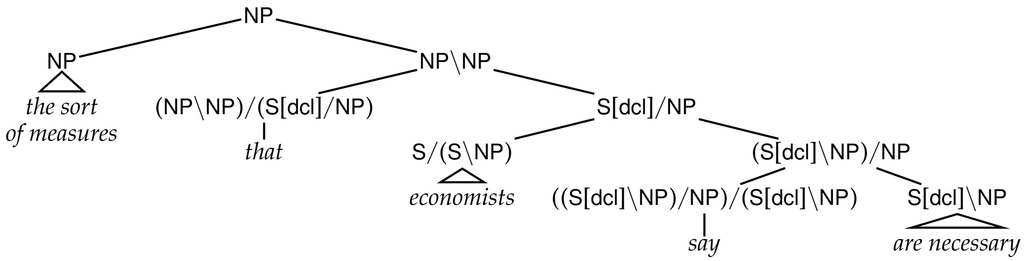
For our translation algorithm, the *T* traces are essential: They indicate the presence of a long-range dependency for a particular argument of the verb, and allow us to use a mechanism similar to GPSG's slash-feature passing (Gazdar et al. 1985), so that long-range dependencies are represented in the gold-standard dependency structures of the test and training data. This is crucial to correctly inducing and evaluating grammars and parsers for any expressive formalism, including TAG, GPSG, HPSG, LFG, and MPG. A detailed description of this mechanism and of our treatment of other constructions that use *T* traces can be found in the CCGbank manual.

This algorithm works also if there is a coordinate structure within the relative clause such that there are two *T* traces (*the interest rates they pay* *T* *on their deposits and charge* *T* *on their loans*), resulting in the following long-range dependencies:

$$\langle \textit{pay}, \mathsf{((S[dcl]\backslash NP)/PP)/NP}, 3, \textit{rates}\rangle,\ \langle \textit{charge}, \mathsf{((S[dcl]\backslash NP)/PP)/NP}, 3, \textit{rates}\rangle \tag{34}$$

6.2.1 Subject Extraction from Embedded Sentences.

In CCG, verbs which take a bare sentential complement have the category $((\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}_i)/(\mathsf{S}\backslash\mathsf{NP}_i)$ if the subject of the sentential complement is extracted (Steedman 1996). In order to obtain these categories from the Treebank (where the corresponding subject trace is in its canonical position), we assume that the verb takes the VP and the NP argument in reversed order and change the tree accordingly before translation, resulting in the correct CCG analysis:



We obtain the following long-range dependencies:

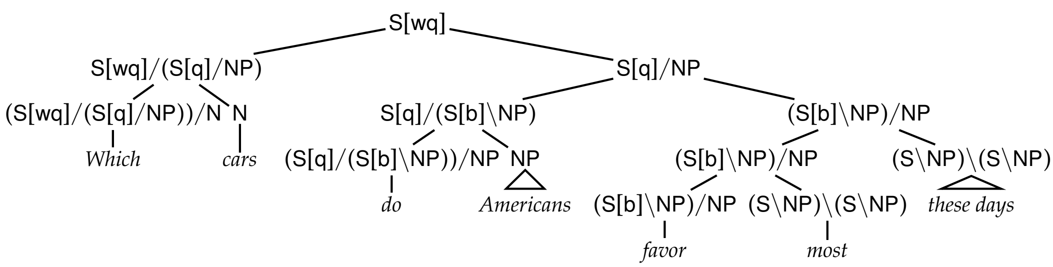
$$\langle \mathit{are}, ((\mathsf{S[dcl]}\backslash\mathsf{NP})/(\mathsf{S[adj]}\backslash\mathsf{NP})), 1, \mathit{sort}\rangle,\ \langle \mathit{necessary}, \mathsf{S[adj]}\backslash\mathsf{NP}, 1, \mathit{sort}\rangle \qquad (35)$$

Because our grammar does not use Baldridge's (2002) modalities or Steedman's (1996) equivalent rule-based restrictions, which prohibit this category from applying to in situ NPs, this may lead to overgeneralization. However, such examples are relatively frequent: There are 97 instances of ((S[.]\NP)/NP)/(S[dcl]\NP) in sections 02–21, and to omit this category would reduce coverage and recovery of long-range extractions.

6.2.2 Wh-*Questions.* *T* traces are also used for *wh*-questions:

```
(SBARQ (WHNP-1 (WDT Which) (NNS cars))                    (36)
       (SQ (VBP do)
           (NP-SBJ Americans)
           (VP (VB favor)
               (NP (-NONE- *T*-1))
               (ADVP most)
               (NP-TMP these days)))
       (. ?)))
```

By percolating the *T* trace up to the SQ-level in a similar way to relative clauses and treating *Which* as syntactic head of the WHNP, we obtain the desired CCG analysis:



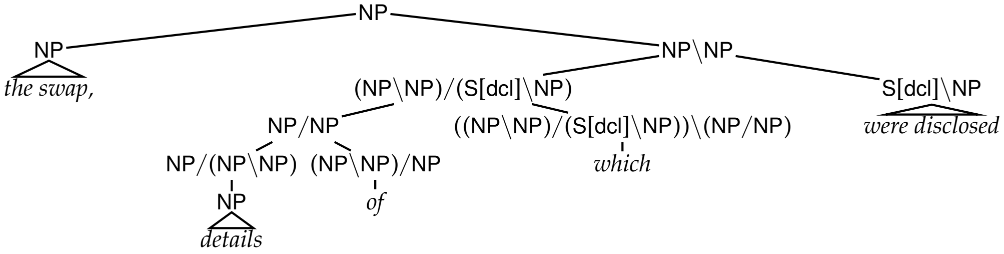
We coindex the head of the extracted NP with that of the noun (*cars*): $(\mathsf{S[wq]}/(\mathsf{S[q]}/\mathsf{NP}_i))/\mathsf{N}_i$, and the subject of *do* with the subject of its complement $((\mathsf{S[q]}/(\mathsf{S[b]}\backslash\mathsf{NP}_1))/\mathsf{NP}_i)$ to obtain the following dependencies:

$$\langle \textit{do}, \mathsf{((S[q]/(S[b]\backslash NP))/NP)}, 1, \textit{favor}\rangle,\ \langle \textit{do}, \mathsf{((S[q]/(S[b]\backslash NP))/NP)}, 2, \textit{Americans}\rangle,\ \langle \textit{favor}, \mathsf{((S[b]\backslash NP)/NP)}, 2, \textit{Americans}\rangle,\ \langle \textit{favor}, \mathsf{((S[b]\backslash NP)/NP)}, 2, \textit{cars}\rangle \tag{37}$$

6.2.3 Pied-Piping. `*T*` traces are also used for pied-piping:

```
(NP-SBJ (NP the swap)                                                    (38)
        (, ,)
        (SBAR (WHNP-2 (WHNP (NNS details))
                            (WHPP (IN of) (WHNP (WDT which))))
              (S *T*-2 were disclosed)))
```

In this example, we need to rebracket the Treebank tree so that *details of* forms a constituent,[18] apply a special rule to assign the category (NP\NP)/NP to the preposition, and combine it via type-raising and composition with *details*. This constituent is then treated as an argument of the relative pronoun:



With appropriate coindexation $\mathsf{((NP\backslash NP_i)/(S[dcl]\backslash NP_j))\backslash(NP/NP_i)_j}$, we obtain the following non-local dependencies:[19]

$$\langle \textit{of}, \mathsf{(NP\backslash NP)/NP}, 1, \textit{details}\rangle,\ \langle \textit{of}, \mathsf{(NP\backslash NP)/NP}, 2, \textit{swap}\rangle,\ \langle \textit{were}, \mathsf{(S[dcl]\backslash NP)/(S[pss]\backslash NP)}, 1, \textit{details}\rangle,\ \langle \textit{disclosed}, \mathsf{S[pss]\backslash NP}, 1, \textit{details}\rangle \tag{39}$$

6.2.4 Extraction of Adjuncts. `*T*` traces can also stand for extracted adjuncts:

```
(S (SBAR-TMP (WHADVP-1 (WRB When))                                      (40)
             (S (NP-SBJ the stock market)
                (VP (VBD dropped)
                    (ADVP-TMP (-NONE- *T*-1)))))
   (S the Mexico fund plunged about 18%))
```

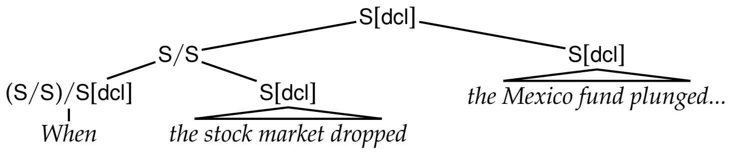
Because adjuncts generally do not extract unboundedly,[20] the corresponding traces (which account for 20% of all `*T*` traces) can be ignored by the translation procedure.

[18] In cases where the WHPP is not enclosed in another noun phrase (*the swap, in which details were disclosed*), no preprocessing is required to obtain the desired CCG derivation.

[19] The index *j* requires that the lexical head of the NP/NP (*details of*) is *details*.

[20] Unbounded extractions are only allowed in combination with certain verbs, for example: *The year in which Ms Fazool says that she was born.* The current lexicon for these verbs does not support the correct analysis of such extractions.

Instead, the dependency between *when* and *dropped* is directly established by the fact that *dropped* is the head of the complement S[dcl]:



This results in the following set of dependencies of *when*:

$$\langle \textit{When}, \mathsf{(S/S)/S[dcl]}, 1, \textit{plunged}\rangle,\quad \langle \textit{When}, \mathsf{(S/S)/S[dcl]}, 2, \textit{dropped}\rangle \tag{41}$$

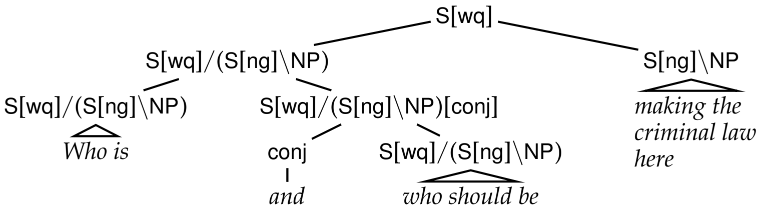
6.3 Long-Range Dependencies Through Coordination

6.3.1 Right Node Raising. Just as composition and type-raising permit CCG analyses of *wh*-extraction, which use the same lexical categories as for in situ complements, they also provide an analysis of right node raising constructions without introducing any new lexical categories.

In the Treebank analysis of right node raising, the shared constituent is coindexed with two *RNR* traces in both of its canonical positions:

```
(SBARQ (SBARQ (WHNP-5 (WP Who))                                    (42)
              (SQ (NP-SBJ (-NONE- *T*-5))
                  (VP (VBZ is)
                      (VP (-NONE- *RNR*-4)))))
       (CC and)
       (SBARQ (WHNP-6 (WP who))
              (SQ (NP-SBJ (-NONE- *T*-6))
                  (VP (MD should)
                      (VP (VB be)
                          (VP (-NONE- *RNR*-4))))))
       (VP-4 (VBG making)
             (NP the criminal law)
             (ADVP-LOC (RB here)))
       (. ?))
```

We need to alter the translation algorithm slightly to deal with *RNR* traces in a manner essentially equivalent to the earlier treatment of *T* *wh*-traces. Details are in the CCGbank manual. The CCG derivation for the above example is as follows:



The right node raising dependencies are as follows:

$$\langle \textit{is}, \mathsf{(S[dcl]\backslash NP)/(S[ng]\backslash NP)}, 2, \textit{making}\rangle, \\ \langle \textit{be}, \mathsf{(S[b]\backslash NP)/(S[ng]\backslash NP)}, 2, \textit{making}\rangle \tag{43}$$

Our algorithm works also if the shared constituent is an adjunct, or if two conjoined noun phrases share the same head, which is also annotated with *RNR* traces.

Although there are only 209 sentences with *RNR* traces in the entire Treebank, right node raising is actually far more frequent, because *RNR* traces are not used when the conjuncts consist of single verb tokens. The Treebank contains 349 VPs in which a verb form (/VB/) is immediately followed by a conjunction (CC) and another verb form, and has an NP sister (without any coindexation or function tag). In CCGbank, sections 02–21 alone contain 444 sentences with verbal or adjectival right node raising.

6.4 Right Node Raising Parasitic Gaps

Right node raising is also marked in the Penn Treebank using *RNR* traces for "parasitic gap" constructions such as the following:

```
a. (VP (VBN held)                                                  (44)
       (S (NP-SBJ (-NONE- *-2))
          (VP (TO to)
              (VP (VP (VB cause)
                          (NP (-NONE- *RNR*-1)))
                  (PRN (, ,)
                       (PP (RB rather)
                           (IN than)
                           (VP (VB resolve)
                               (NP (-NONE- *RNR*-1))))
                       (, ,))
                  (NP-1 (NN conflict))))))
b. (S (NP-SBJ These first magnitude wines)
      (VP (VBD ranged)
          (PP-CLR-LOC in price)
          (PP-DIR (PP (IN from)
                      (NP (NP $40)
                          (NP-ADV (-NONE- *RNR*-1))))
                  (PP (TO to)
                      (NP (NP $125)
                         (NP-ADV (-NONE- *RNR*-1))))
                  (NP-ADV-1 a bottle))))
```

These sentences require rules based on the substitution combinator **S** (Steedman 1996). Our treatment of right node raising traces deals with the first case correctly, via the backward crossing rule $<\mathbf{S}_{\times}$, and allows us to obtain the following correct dependencies:

$$\begin{array}{l}\langle \textit{cause}, ((\mathsf{S[b]}\backslash\mathsf{NP})/\mathsf{NP}), 1, \textit{system}\rangle{,}1\ \langle \textit{cause}, ((\mathsf{S[b]}\backslash\mathsf{NP})/\mathsf{NP}), 2, \textit{conflict}\rangle, \\ \langle \textit{resolve}((\mathsf{S[b]}\backslash\mathsf{NP})/\mathsf{NP}), 1, \textit{system}\rangle, \langle \textit{resolve}((\mathsf{S[b]}\backslash\mathsf{NP})/\mathsf{NP}), 2, \textit{conflict}\rangle\end{array} \quad (45)$$

The second type of parasitic gap, (44b), would be handled equally correctly by the forward substitution rule $>\mathbf{S}$, since the PPs are both arguments. Unfortunately, as we saw in Section 3, the Treebank classifies such PPs as directional adverbials, hence we translate them as adjuncts and lose such examples, of which there are at least three more, all also involving *from* and *to*:

a. *Home purchase plans have ranged monthly from 2.9% to 3.7% of respondents.* (46)
b. *They'll go from being one of the most leveraged to one of the least leveraged casino companies.*
c. *the decline in average gold price realization to $367 from $429 per ounce*

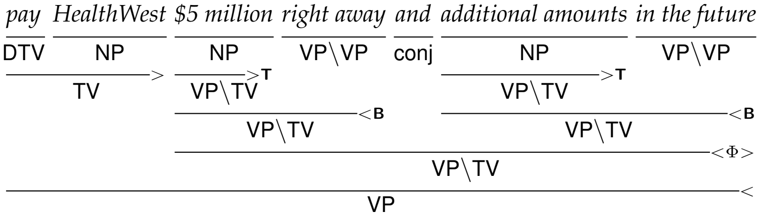
As in the case of leftward extraction, including such long-range dependencies in the dependency structure is crucial to correct induction and evaluation of all expressive grammar formalisms. Although no leftward-extracting parasitic gaps appear to occur in the Treebank, our grammar and model predicts examples like the following, and will cover them when encountered:

Conflict which the system was held to cause, rather than resolve. (47)

6.4.1 Argument Cluster Coordination. If two VPs with the same head are conjoined, the second verb can be omitted. The Treebank encodes these constructions as a VP-coordination in which the second VP lacks a verb. The daughters of the second conjunct are coindexed with the corresponding elements in the first conjunct using a = index:

```
(VP (VP (VB pay)                                (48)
        (NP HealthVest)
        (NP-2 $ 5 million)
        (ADVP-TMP-3 right away))
    (CC and)
    (VP (NP=2 additional amounts)
        (PP-TMP=3 in the future))
```

In the CCG account of this construction, *$5 million right away* and *additional amounts in the future* form constituents ("argument clusters"), which are then coordinated. These constituents are obtained by type-raising and composing the arguments in each conjunct, yielding a functor which takes a verb with the appropriate category to its left to yield a verb phrase (Dowty 1988; Steedman 1985). Then the argument clusters are conjoined, and combine with the verb via function application:[21]



[21] We use the following abbreviations: VP for S\NP, TV for transitive (S\NP)/NP, and DTV for ditransitive ((S\NP)/NP/NP).

This construction is one in which the CCGbank head-dependency structure (shown subsequently) fails to capture the full set of predicate–argument structure relations that would be implicit in a full logical form:

$$\begin{array}{l}\langle \textit{pay}, \texttt{((S[b]\backslash NP)/NP)/NP}, 3, \$\rangle, \langle \textit{pay}, \texttt{((S[b]\backslash NP)/NP)/NP}, 3, \textit{amounts}\rangle, \\ \langle \textit{away}, \texttt{(S\backslash NP)\backslash(S\backslash NP)}, 2, \textit{pay}\rangle, \langle \textit{in}, \texttt{((S\backslash NP)\backslash(S\backslash NP))/NP}, 2, \textit{pay}\rangle\end{array} \quad (50)$$

That is, the dependency structure does not express the fact that *right away* takes scope over *$5 million* and *in future* over *additional amounts*, rather than the other way around. However, this information is included in the full surface-compositional semantic interpretation that is built by the combinatory rules.

Because the Treebank constituent structure does not correspond to the CCG analysis, we need to transform the tree before we can translate it. During preprocessing, we create a copy of the entire argument cluster which corresponds to the constituent structure of the CCG analysis. During normal category assignment, we use the first conjunct in its original form to obtain the correct categories of all constituents. In a later stage, we use type-raising and composition to combine the constituents within each argument cluster. For a detailed description of this algorithm and a number of variations on the original Treebank annotation that we did not attempt to deal with, the interested reader is referred to the CCGbank manual.

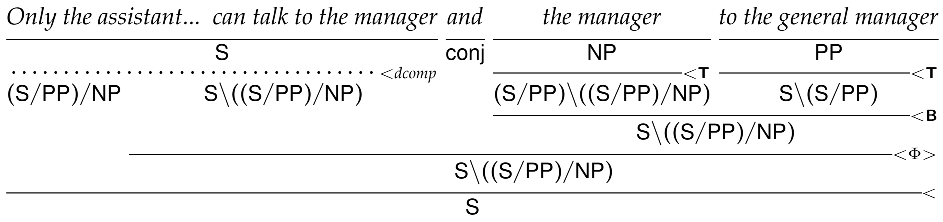
There are 226 instances of argument-cluster coordination in the entire Penn Treebank. The algorithm delivers a correct CCG derivation for 146 of these. Translation failures are due to the fact that the algorithm can at present only deal with this construction if the two conjuncts are isomorphic in structure, which is not always the case. This is unfortunate, because CCG is particularly suited for this construction. However, we believe that it would be easier to manually reannotate those sentences that are not at present translated than to try to adapt the algorithm to deal with all of them individually.

6.4.2 Gapping. For sentential gapping, the Treebank uses annotation similar to argument cluster coordination:

```
(S (S (NP-SBJ-1 Only the assistant manager)                (51)
      (VP (MD can)
          (VP (VB talk)
              (PP-CLR-2 to the manager))))
   (CC and)
   (S (NP-SBJ=1 the manager)
      (PP-CLR=2 to the general manager)))
```

This construction cannot be handled with the standard combinatory rules of CCG that are assumed for English. Instead, Steedman (2000) proposes an analysis of gapping that uses a unification-based "decomposition" rule. Categorial decomposition allows a category type to be split apart into two subparts, and is used to yield an analysis of gapping that is very similar to that of argument cluster coordination:[22]

[22] It is only the syntactic types that are decomposed or recovered in this way: the corresponding semantic entities and in particular the interpretation for the gapped verb group *can talk* must be available from the left conjunct's information structure, via anaphora. That is, decomposition adds very little to the categorial information available from the right conjunct, except to make the syntactic types yield an S. The real work is done in the semantics.



(52)

Because the derivation is not a tree anymore, and the decomposed constituents do not correspond to actual constituents in the surface string, this analysis is difficult to represent in a treebank. The 107 sentences that contain sentential gapping are therefore omitted in the current version of CCGbank, even though special coordination rules that mimic the decomposition analysis are conceivable.

6.5 Other Null Elements in the Treebank

Besides the cases discussed herein, the Treebank contains further kinds of null elements, all of which the algorithm ignores. The null element `*ICH*` ("Insert Constituent Here"), which appears 1,240 times, is used for extraposition of modifiers. Like ellipsis, this is a case of a semantic dependency which we believe to be anaphoric, and therefore not reflected in the syntactic category. For this reason we treat any constituent that is coindexed with an `*ICH*` as an adjunct. The null element `*PPA*` ("Permanent Predictable Ambiguity," 26 occurrences) is used for genuine attachment ambiguities. Since the Treebank manual states that the actual constituent should be attached at the more likely attachment site, we chose to ignore any `*PPA*` null element. Our algorithm also ignores the null element `*?*`, which occurs 582 times, and indicates "a missing predicate or a piece thereof" (Marcus, Santorini, and Marcinkiewicz 1993). It is used for VP ellipsis, and can also occur in conjunction with a VP pro-form *do* (*You either believe he can do it or you don't* `*?*`), or in comparatives (*the total was far higher than expected* `*?*`).[23]

6.6 The Complete Translation Algorithm

We can now define the complete translation algorithm, including the modifications necessary to deal with traces and argument clusters:

```
foreach tree τ:
    preprocessTree(τ);
    determineConstituentTypes(τ);
    makeBinary(τ);
    percolateTraces(τ);
    assignCategories(τ);
    treatArgumentClusters(τ);
    cutTracesAndUnaryRules(τ);
    verifyDerivation(τ);
    assignDependencies(τ);
```

[23] We believe that both conjuncts in the first example are complete sentences which are related anaphorically. Therefore, the syntactic category of *do* is S[dcl]\NP, not (S[dcl]\NP)/VP. In the second example, `*?*` indicates a semantic argument of *expected* that we do not reflect in the syntactic category.

The successive steps have the following more detailed character:

preprocessTree: Correct tagging errors, ensure the constituent structure conforms to the CCG analysis. Eliminate quotes. Create copies of coordinated argument clusters that correspond to the CCG analysis.

determineConstituentTypes: For each node, determine its constituent type (head, complement, adjunct, conjunction, a constituent that is coindexed with a `*RNR*` trace, spurious null element, or argument cluster).

makeBinary: Binarize the tree.

percolateTraces: Determine the CCG category of `*T*` and `*RNR*` traces in complement position, and percolate them up to the appropriate level in the tree.

assignCategories: Assign CCG categories to nodes in the tree, starting at the root node. Nodes that are coindexed with `*RNR*` traces receive the category of the corresponding traces. Argument clusters are ignored in this step.

treatArgumentClusters: Assign categories to argument clusters.

cutTracesAndUnaryRules: Cut out constituents that are not part of the CCG derivation, such as traces, null elements, and the copy of the first conjunct in argument cluster coordination. Eliminate resulting unary projections of the form $\mathtt{X} \Rightarrow \mathtt{X}$.

verifyDerivation: Discard those trees for which the algorithm does not produce a valid CCG derivation. In most cases, this is due to argument cluster coordination that is not annotated in a way that our algorithm can deal with.

assignDependencies: coindex specific classes of lexical categories to project non-local dependencies, and generate the word–word dependencies that constitute the underlying predicate–argument structure.

7. Remaining Problems for the Translation Algorithm

In a number of cases, missing structure or a necessary distinction between different constructions needed to inform the translation is missing, and cannot be inferred deterministically from the Treebank analysis without further manual re-annotation. We discuss these residual problems here, because they are likely to present obstacles to the extraction of linguistically adequate grammars in any formalism.

7.1 Complement/Adjunct Distinction

Our translation algorithm requires a distinction between complements and adjuncts. In many cases, this distinction is easily read off the Treebank annotation, but it is in general an open linguistic problem (McConnell-Ginet 1982). Because the Treebank annotation does not explicitly distinguish between complements and adjuncts, researchers typically develop their own heuristics—see, for example, Kinyon and Prolo (2002). For prepositional phrases, we rely on the `-CLR` ("closely related") function tag to identify complements, although it is unclear whether the Treebank annotators were able to use this tag consistently. Not all PP arguments seem to have this function tag, and some PPs that have this tag may have been better considered adjuncts:

```
a. (VP (VBN replaced)                                  (53)
       (NP (-NONE- *-1))
       (PP with a different kind of filter))
```

b.
```
(VP (VB redeploy)
    (NP their money)
    (PP-CLR at lower rates))
```

For TAG, Chen, Bangalore, and Vijay-Shanker (2006) show that different heuristics yield grammars that differ significantly in size, coverage, and linguistic adequacy. We have not attempted such an investigation. In a future version of CCGbank, it may be possible to follow Shen and Joshi (2005) in using the semantic roles of the Proposition Bank (Palmer, Gildea, and Kingsbury 2005) to distinguish arguments and adjuncts.

7.2 Phrasal Verbs

Particle-verb constructions are difficult to identify in the Treebank, because particles can be found as PRT, ADVP-CLR, and ADVP. Therefore, verbs in the CCGbank grammar do not subcategorize for particles, which are instead treated as adverbial modifiers.

7.3 Compound Nouns

Compound nouns are often inherently ambiguous, and in most cases, the Treebank does not specify their internal structure:

```
(NP (JJ only) (JJ French) (NN history) (NNS questions))
```
(54)

In order to obtain the correct analysis, manual re-annotation would be required. Because this was not deemed feasible within our project, compound nouns are simply translated into strictly right-branching binary trees, which yields the correct analysis in some, but not all, cases. This eschews the computational problem that a grammar for compound nouns induces all possible binary bracketings, but is linguistically incorrect.

7.4 Coordinate Nouns

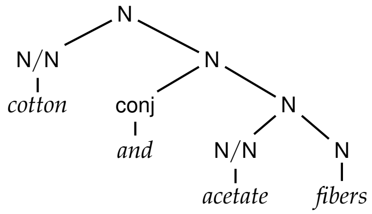
A similar problem arises in compound nouns that involve internal coordination:

```
(NP (NN cotton) (CC and) (NN acetate) (NNS fibers))
```
(55)

We include the following (linguistically incorrect) rule in our grammar, which yields a default dependency structure corresponding to N/N coordination:

$$\text{conj} \quad \mathsf{N} \Rightarrow \mathsf{N} \qquad (56)$$

This rule allows us to translate the above tree as follows:

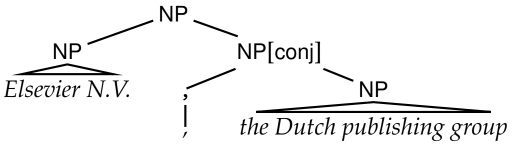


7.5 Appositives and Lists

The Treebank markup of NP appositives is indistinguishable from that of NP lists:

```
(NP (NP Elsevier N.V.)
    (, ,)
    (NP the Dutch publishing group))
```
(57)

Therefore, our current grammar does not distinguish between appositives and NP coordination, even though appositives should be analyzed as predicative modifiers. This leads to a reduction of ambiguity in the grammar, but is semantically incorrect:



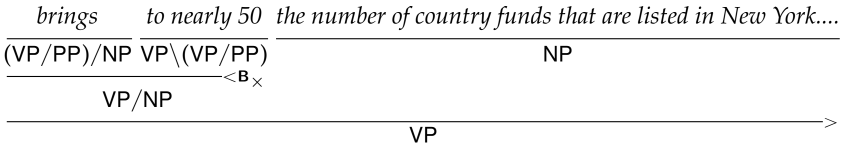
7.6 Lack of Number Agreement

Our current grammar does not implement number agreement (which is, however, represented in the POS tags). One problem that prevented us from including number agreement is the above-mentioned inability to distinguish NP lists and appositives.

7.7 Attachment of Noun Phrase Modifiers

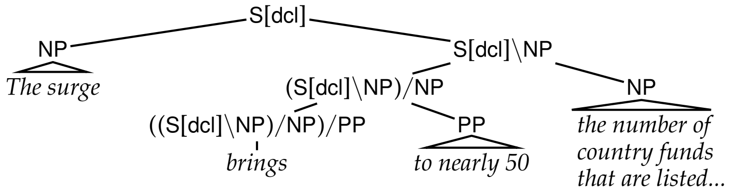
In the Penn Treebank, all relative clauses are attached at the noun phrase level. This is semantically undesirable, because a correct interpretation of restrictive relative clauses can only be obtained if they modify the noun, whereas non-restrictive relative clauses are noun phrase modifiers. Because this distinction requires manual inspection on a case-by-case basis, we were unable to modify the Treebank analysis. Thus, all CCGbank relative pronouns have categories of the form $(\mathrm{NP}_i\backslash\mathrm{NP}_i)/(\mathrm{S}/\mathrm{NP}_i)$, rather than $(\mathrm{N}_i\backslash\mathrm{N}_i)/(\mathrm{S}/\mathrm{NP}_i)$. This will make life difficult for those trying to provide a Montague-style semantics for relative modifiers. Like most other problems that we were not able to overcome, this limitation of the Treebank ultimately reflects the sheer difficulty of providing a consistent and reliable annotation for certain linguistic phenomena, such as modifier scope.

7.7.1 Heavy NP Shift. In English, noun phrase arguments can be shifted to the end of the sentence if they become too "heavy." This construction was studied extensively by Ross (1967). The CCG analysis (Steedman 1996) uses backward crossed composition to provide an analysis where *brings* has its canonical lexical category (VP/PP)/NP:



(58)

Because the Penn Treebank does not indicate heavy NP shift, the corresponding CCGbank derivation does not conform to the desired analysis, and requires additional lexical categories which may lead to incorrect overgeneralizations:[24]



This will also be a problem in using the Penn Treebank or CCGbank for any theory of grammar that treats heavy NP shift as extraction or movement.

8. Coverage, Size, and Evaluation

Here we first examine briefly the coverage of the translation algorithm on the entire Penn Treebank. Then we examine the CCG grammar and lexicon that are obtained from CCGbank. Although the grammar of CCG is usually thought of as consisting only of the combinatory rule schemata such as (3) and (5), we are interested here in the instantiation of these rules, in which the variables X and Y are bound to values such as S and NP, because statistical parsers such as Hockenmaier and Steedman's (2002) or Clark and Curran's (2004) are trained on counts of such instantiations. We report our results on sections 02–21, the standard training set for Penn Treebank parsers, and use section 00 to evaluate coverage of the training set on unseen data. Sections 02–21 contains 39,604 sentences (929,552 words/tokens), whereas section 00 consists of 1,913 sentences (45,422 words/tokens).

8.1 Coverage of the Translation Algorithm

CCGbank contains 48,934 (99.44%) of the 49,208 sentences in the entire Penn Treebank. The missing 274 sentences could not be automatically translated to CCG. This includes 107 instances of sentential gapping, a construction our algorithm does not cover (see Section 6.4.2), and 66 instances of non-sentential gapping, or argument-cluster coordination (see Section 6.4.1).

The remaining translation failures include trees that consist of sequences of NPs that are not separated by commas, some fragments, and a small number of constructions involving long-range dependencies, such as *wh*-extraction, parasitic gaps, or argument cluster coordinations where the translation did not yield a valid CCG derivation because a complement had been erroneously identified as an adjunct.

[24] Backward crossed composition is also used by Steedman (1996, 2000) and Baldridge (2002) to account for constraints on preposition stranding in English. Because this rule in its unrestricted form leads to overgeneralization, Baldridge restricts crossing rules via the × modality. The current version of CCGbank does not implement modalities, but because the grammar that is implicit in CCGbank only consists of particular seen rule instantiations, it may not be affected by such overgeneration problems.

Table 1
The 20 tokens with the highest number of lexical categories and their frequency (sections 02-21).

Word	#Cats.	Freq.	Word	#Cats.	Freq.
as	130	4237	*of*	59	22782
is	109	6893	*that*	55	7951
to	98	22056	*-LRB-*	52	1140
than	90	1600	*not*	50	1288
in	79	15085	*are*	48	3662
–	67	2001	*with*	47	4214
's	67	9249	*so*	47	620
for	66	7912	*if*	47	808
at	63	4313	*on*	46	5112
was	61	3875	*from*	46	4437

8.2 The Lexicon

A CCG lexicon specifies the lexical categories of words, and therefore contains the entire language-specific grammar. Here, we examine the size and coverage of the lexicon that consists of the word–category pairs that occur in CCGbank. This lexicon could be used by any CCG parser, although morphological generalization (which is beyond the scope of the present paper) and ways to treat unknown words are likely to be necessary to obtain a more complete lexicon.

Number of Entries. The lexicon extracted from sections 02–21 has 74,669 entries for 44,210 word types (or 929,552 word tokens). Many words have only a small number of categories, but because a number of frequent closed-class items have a large number of categories (see Table 1), the expected number of lexical categories per token is 19.2.

Number and Growth of Lexical Category Types. How likely is it that we have observed the complete inventory of category types in the English language? There are 1,286 lexical category types in sections 02–21. Figure 4 examines the growth of the number of lexical category types as a function of the amount of data translated into CCG. The log–log plot

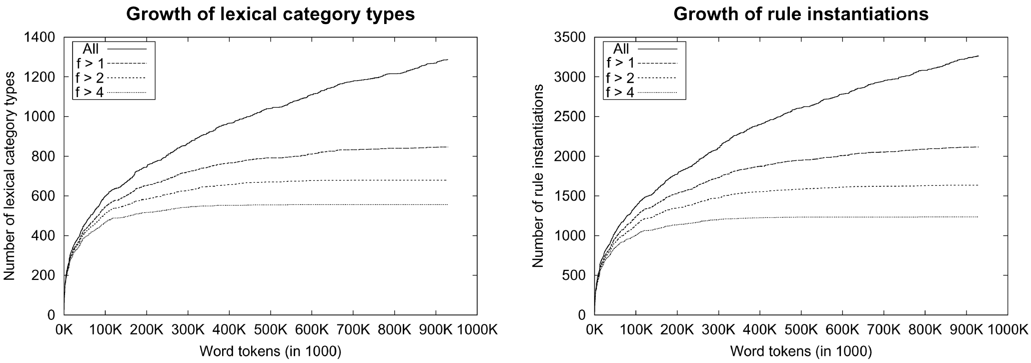


Figure 4
The growth of lexical category types and rule instantiations (sections 02–21).

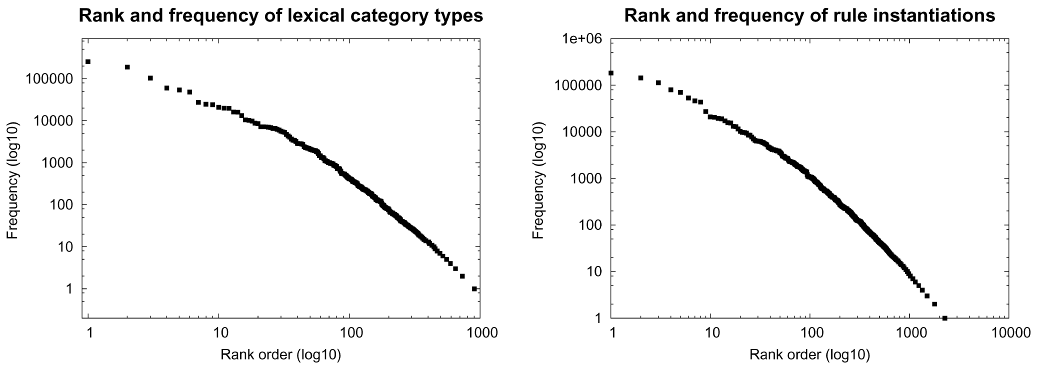


Figure 5
A log–log plot of the rank order and frequency of the lexical category types (left) and instantiations of combinatory rules (right) in CCGbank.

of the rank order and frequency of the lexical categories in Figure 5 indicates that the underlying distribution is roughly Zipfian, with a small number of very frequent categories and a long tail of rare categories. We note 439 categories that occur only once, and only 556 categories occur five times or more. Inspection suggests that although some of the category types that occur only once are due to noise or annotation errors, most are correct and are in fact required for certain constructions. Typical examples of rare but correct and necessary categories are relative pronouns in pied-piping constructions, or verbs which take expletive subjects.

Lexical Coverage on Unseen Data. The lexicon extracted from sections 02–21 contains the necessary categories (as determined by our translation algorithm) for 94.0% of all tokens in section 00 (42,707 out of 45,422). The missing entries that would be required for the remaining 6% of tokens fall into two classes: 1,728, or 3.8%, correspond to completely unknown words that do not appear at all in section 02–21, whereas the other 2.2% of tokens do appear in the training set, but not with the categories required in section 00.

All statistical parsers have to be able to accept unknown words in their input, regardless of the underlying grammar formalism. Typically, frequency information for rare words in the training data is used to estimate parameters for unknown words (and when these rare or unknown words are encountered during parsing, additional information may be obtained from a POS-tagger (Collins 1997)). However, in a lexicalized formalism such as CCG, there is the additional problem of missing lexical entries for known words. Because lexical categories play such an essential role in CCG, even a small fraction of missing lexical entries can have a significant effect on coverage, since the parser will not be able to obtain the correct analysis for any sentence that contains such a token. Hockenmaier and Steedman (2002) show that this lexical coverage problem does in practice have a significant impact on overall parsing accuracy. However, because many of the known words with missing entries do not appear very often in the training data, Hockenmaier (2003a) demonstrates that this problem can be partially alleviated if the frequency threshold below which rare words are treated as unseen is set to a much higher value than for standard Treebank parsers. An alternative approach, advocated by Clark and Curran (2004), is to use a supertagger which predicts lexical CCG categories in combination with a discriminative parsing model.

8.3 The Syntactic Component

Size and Growth of Instantiated Syntactic Rule Set. Statistical CCG parsers such as Hockenmaier and Steedman (2002) or Clark and Curran (2004) are trained on counts of specific instantiations of combinatory rule schemata by category-types. It is therefore instructive to consider the frequency distribution of these category-instantiated rules.

The grammar for sections 02-21 has 3,262 instantiations of general syntactic combinatory rules like those in (3) with specific categories. Of these, 1146 appear only once, and 2,027 appear less than five times. Although there is some noise, many of the CCG rules that appear only once are linguistically correct and should be used by the parser. They include certain instantiations of type-raising, coordination, or punctuation rules, or rules involved in argument cluster coordinations, pied-piping constructions, or questions, all of which are rare in the *Wall Street Journal*. As can be seen from Figure 5, the distribution of rule frequencies is again roughly Zipfian, with the 10 most frequent rules accounting for 59.2% of all rule instantiations (159 rules account for 95%; 591 rules for 99%). The growth of rule instantiations is shown in Figure 4. If function tags are ignored, the grammar for the corresponding sections of the original Treebank contains 12,409 phrase-structure rules, out of which 6,765 occur only once (Collins 1999). These rules also follow a Zipfian distribution (Gaizauskas 1995). The fact that both category types and rule instances are also Zipfian for CCGbank, despite its binarized rules, shows that the phenomenon is not just due to the Treebank annotation with its very flat rules.

Syntactic Rule Coverage on Unseen Data. Syntactic rule coverage for unseen data is almost perfect: 51,932 of the 51,984 individual rule instantiations in section 00 (corresponding to 844 different rule types) have been observed in section 02–21. Out of the 52 missing rule instantiation tokens (corresponding to 38 rule types, because one rule appears 13 times in one sentence), six involve coordination, and three punctuation. One missing rule is an instance of substitution (caused by a parasitic gap). Two missing rules are instances of type-raised argument types combining with a verb of a rare type.

9. Conclusion

This paper has presented an algorithm which translates Penn Treebank phrase-structure trees into CCG derivations augmented with word–word dependencies that approximate the underlying predicate–argument structure. In order to eliminate some of the noise in the original annotation and to obtain linguistically adequate derivations that conform to the "correct" analyses proposed in the literature, considerable preprocessing was necessary. Even though certain mismatches between the syntactic annotations in the Penn Treebank and the underlying semantics remain, and will affect any similar attempt to obtain expressive grammars from the Treebank, we believe that CCGbank, the resulting corpus, will be of use to the computational linguistics community in the following ways.

CCGbank has already enabled the creation of several robust and accurate wide-coverage CCG parsers, including Hockenmaier and Steedman (2002), Clark, Hockenmaier, and Steedman (2002), Hockenmaier (2003b), and Clark and Curran (2004, 2007). Although the construction of full logical forms was beyond the scope of this project, CCGbank can also be seen as a resource which may enable the automatic construction of full semantic interpretations by wide-coverage parsers. Unlike most Penn Treebank parsers, such as Collins (1999) or Charniak (2000), these CCGbank parsers return not only syntactic derivations, but also local and long-range dependencies, including those that arise under relativization and coordination. Although these dependencies are only an approximation of the full semantic interpretation that can in principle be obtained from a CCG, they may prove useful for tasks such as summarization and question answering (Clark, Steedman, and Curran 2004). Furthermore, Bos et al. (2004) and Bos (2005) have demonstrated that the output of CCGbank parsers can be successfully translated into Kamp and Reyle's (1993) Discourse Representation Theory structures, to support question answering and the textual entailment task (Bos and Markert 2005).

We hope that these results can be ported to other corpora and other similarly expressive grammar formalisms. We also hope that our experiences will be useful in designing guidelines for future treebanks. Although implementational details will differ across formalisms, similar problems and questions to those that arose in our work will be encountered in any attempt to extract expressive grammars from annotated corpora.

Because CCGbank preserves most of the linguistic information in the Treebank in a somewhat less noisy form, we hope that others will find it directly helpful for inducing grammars and statistical parsing models for other linguistically expressive formalisms. There are essentially three ways in which this might work.

For lexicalized grammars, it may in some cases be possible to translate the subcategorization frames in the CCG lexicon directly into the target theory. For type-logical grammars (Moortgat 1988; Morrill 1994; Moot 2003), this is little more than a matter of transducing the syntactic types for the lexicon into the appropriate notation. For formalisms like LTAG, the relation is more complex, but the work of Joshi and Kulick (1996), who "unfold" CCG categories into TAG elementary trees via partial proof trees, and Shen and Joshi (2005), who define LTAG "spines" that resemble categories, suggest that this is possible. Transduction into HPSG signs is less obvious, but also seems possible in principle.

A second possibility is to transduce CCGbank itself into a form appropriate to the target formalism. There seems to be a similar ordering over alternative formalisms from straightforward to less straightforward for this approach. We would also expect that dependency grammars Mel'čuk and Pertsov 1987; Hudson 1984) and parsers (McDonald, Crammer, and Pereira 2005) could be trained and tested with little extra work on the dependencies in CCGbank.

Finally, we believe that existing methods for translating the Penn Treebank from scratch into other grammar formalisms will benefit from including preprocessing similar to that described here.

As some indication of the relative ease with which these techniques transfer, we offer the observation that the 900K-word German Tiger dependency corpus has recently been translated into CCG using very similar techniques by Hockenmaier (2006), and Çakıcı (2005) has derived a Turkish lexicon from the a similarly preprocessed version of the METU-Sabançı Turkish dependency treebank (Oflazer et al. 2003).

A fundamental assumption behind attempts at the automatic translation of syntactically annotated corpora into different grammatical formalisms such as CCG, TAG, HPSG, or LFG is that the analyses that are captured in the original annotation can be mapped directly (or, at least, without too much additional work) into the desired analyses in the target formalism. This can only hold if all constructions that are treated in a similar manner in the original corpus are also treated in a similar manner in the target formalism. For the Penn Treebank, our research and the work of others (Xia 1999; Chen and Vijay-Shanker 2004; Chiang 2000; Cahill et al. 2002) have shown that such a correspondence exists in most cases.

Although the output of most current Treebank parsers is linguistically impoverished, the Treebank annotation itself is not. It is precisely the linguistic richness and detail of the original annotation—in particular, the additional information present in the null elements and function tags that are ignored by most other parsers—that has made the creation of CCGbank possible. The translation process would have been easier if some of the annotation had been more explicit and precise (as in the case of VP coordination, where preprocessing was required to identify the conjuncts, or in NP coordination, where we were not able to distinguish NP lists from appositives) and consistent (most importantly in identifying adjuncts and arguments). An important conclusion that follows for the builders of future treebanks is that the tradition established by the Penn Treebank of including all linguistically relevant dependencies should be continued, with if anything even closer adherence to semantically informed linguistic insights into predicate–argument structural relations. Our results also indicate that corpora of at least the order of magnitude of the Penn Treebank are necessary to obtain grammars and parsers that are sufficiently expressive, robust, and wide in coverage to recover these relations completely.

Acknowledgments

We would like to thank our colleagues in Edinburgh and Philadelphia—in particular Jason Baldridge, Johan Bos, Stephen Clark, James Curran, Michael White, Mitch Marcus, Ann Bies, Martha Palmer, and Aravind Joshi—for numerous conversations and feedback on the corpus. We would also like to thank the Linguistic Data Consortium for their help in publishing CCGbank, and the *Computational Linguistics* reviewers for their extensive comments on earlier versions of this paper.
We gratefully acknowledge the financial support provided by EPSRC grant GR/M96889. JH also acknowledges support by an EPSRC studentship and the Edinburgh Language Technology Group, and by NSF ITR grant 0205456 at the University of Pennsylvania. MJS acknowledges support from the Scottish Enterprise Edinburgh–Stanford Link (NSF IIS-041628 (R39058)) and EU IST grant PACOPLUS (FP6-2004-IST-4-27657).

References

Ajdukiewicz, Kazimierz. 1935. Die syntaktische Konnexität. In Storrs McCall, editor, *Polish Logic 1920–1939*. Oxford University Press, Oxford, pages 207–231. Translated from *Studia Philosophica*, 1, 1–27.

Aone, Chinatsu and Kent Wittenburg. 1990. Zero morphemes in Unification-based Combinatory Categorial Grammar. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 188–193, Pittsburgh, PA.

Bach, Emmon. 1976. An extension of classical transformational grammar. In *Problems in Linguistic Metatheory: Proceedings of the 1976 Conference at Michigan State University*, pages 183–224, Lansing, MI.

Baldridge, Jason. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Bar-Hillel, Yehoshua. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Blaheta, Don and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 234–240, Seattle.

Bos, Johan. 2005. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*, pages 42–53, Tilburg, The Netherlands.

Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 1240–1246, Geneva, Switzerland.

Bos, Johan and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, Canada.

Bozsahin, Cem. 1998. Deriving the predicate-argument structure for a free word order language. In *Proceedings of COLING-ACL '98, Montreal*, pages 167–173, Cambridge, MA.

Buszkowski, Wojciech and Gerald Penn. 1990. Categorial grammars determined from linguistic data by unification. *Studia Logica*, 49:431–454.

Butt, Miriam, Tracy Holloway King, Maria-Eugenia Nino, and Frederique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI Publications, Stanford, CA.

Cahill, Aoife, Michael Burke, Ruth O'Donovan, Josef Van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 319–326, Barcelona, Spain.

Cahill, Aoife, Mairead McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic annotation of the Penn Treebank with LFG F-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation - Bootstrapping Annotated Language Data*, pages 8–15, Las Palmas, Spain.

Çakıcı, Ruken. 2005. Automatic induction of a CCG grammar for Turkish. In *ACL 2005 Student Research Workshop*, pages 73–78, Ann Arbor, MI.

Campbell, Richard. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 645–652, Barcelona, Spain.

Carpenter, Bob. 1991. The generative power of Categorial Grammars and Head-driven Phrase Structure Grammars with lexical rules. *Computational Linguistics*, 17(3):301–314.

Carpenter, Bob. 1992. Categorial grammars, lexical rules, and the English predicative. In Robert Levine, editor, *Formal Grammar: Theory and Implementation*. Oxford University Press, Oxford, chapter 3.

Carroll, John, G. Minnen, and E. Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL-99 Workshop on Linguistically Interpreted Corpora (LINC-99)*, pages 35–41, Bergen, Norway.

Charniak, Eugene. 2000. A Maximum-Entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA.

Chen, John, Srinivas Bangalore, and K. Vijay-Shanker. 2006. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*, 12(03):251–299.

Chen, John and K. Vijay-Shanker. 2004. Extraction of TAGs from Treebank. In H. Bunt, J. Caroll, and G. Satta, editors, *New Developments in Parsing Technology*. Springer, Berlin, pages 73–90.

Chiang, David. 2000. Statistical parsing with an automatically extracted Tree Adjoining Grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 456–463, Hong Kong.

Clark, Stephen and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 103–110, Barcelona, Spain.

Clark, Stephen and James R. Curran. 2007. Formalism-Independent Parser Evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255, Prague, Czech Republic.

Clark, Stephen, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 327–334, Philadelphia, PA.

Clark, Stephen, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using CCG. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*, pages 111–118, Barcelona, Spain.

Collins, Michael. 1997. Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain.

Collins, Michael. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, CA.

Copestake, Ann and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC)*, pages 591–600, Athens, Greece.

Curry, Haskell B. and Robert Feys. 1958. *Combinatory Logic*, volume I.

North-Holland, Amsterdam.

Dienes, Peter and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP'03)*, pages 33–40, Sapporo, Japan.

Dienes, Peter and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431–438, Sapporo, Japan.

Dowty, David. 1978. Governed transformations as lexical rules in a Montague grammar. *Linguistic Inquiry*, 9:393–426.

Dowty, David. 1988. Type-raising, functional composition, and non-constituent coordination. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*. Reidel, Dordrecht, pages 153–198.

Eisner, Jason. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86, Santa Cruz, CA.

Gabbard, Ryan, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York, NY.

Gaizauskas, Robert. 1995. Investigations into the grammar underlying the Penn Treebank. Technical Report CS-95-25, Department of Computer Science, University of Sheffield.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalised Phrase Structure Grammar*. Blackwell, Oxford.

Henderson, James. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 95–102, Barcelona, Spain.

Hepple, Mark. 1990. *The Grammar and Processing of Order and Dependency: a Categorial Aproach*. Ph.D. thesis, University of Edinburgh.

Hepple, Mark and Glyn Morrill. 1989. Parsing and derivational equivalence. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 10–18, Manchester, UK.

Hockenmaier, Julia. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Hockenmaier, Julia. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 359–366, Sapporo, Japan.

Hockenmaier, Julia. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, Sydney, Australia.

Hockenmaier, Julia, Gann Bierner, and Jason Baldridge. 2004. Extending the coverage of a CCG System. *Research in Language and Computation*, 2:165–208.

Hockenmaier, Julia and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, PA.

Hockenmaier, Julia and Mark Steedman. 2005. *CCGbank: Users' Manual*. Department of Computer and Information Science Technical Report MS-CIS-05-09. University of Pennsylvania, Philadelphia, PA.

Hoffman, Beryl. 1995. *Computational Analysis of the Syntax and Interpretation of 'Free' Word-order in Turkish*. Ph.D. thesis, University of Pennsylvania. IRCS Report 95-17.

Hudson, Richard. 1984. *Word Grammar*. Blackwell, Oxford.

Jacobson, Pauline. 1992. Flexible Categorial grammars: Questions and prospects. In Robert Levine, editor, *Formal Grammar*. Oxford University Press, Oxford, pages 129–167.

Johnson, Mark. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, PA.

Joshi, Aravind and Seth Kulick. 1996. Partial proof trees as building blocks for a Categorial grammar. *Linguistics and Philosophy*, 20(6):637–667.

Joshi, Aravind and Yves Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In M. Nivat and M. Podelski, editors, *Tree Automata and Languages*. North-Holland, pages 409–432.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

Kang, Beom-Mo. 1995. On the treatment of complex predicates in categorial grammar. *Linguistics and Philosophy*, 18:61–81.

Kaplan, Ronald and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pages 173–281.

Kinyon, Alexandra and Carlos Prolo. 2002. Identifying verb arguments and their syntactic function in the Penn Treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1982–1987, Las Palmas, Spain.

Komagata, Nobo. 1999. *Information Structure in Texts: A Computational Analysis of Contextual Appropriateness in English and Japanese*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

König, Esther. 1994. A hypothetical reasoning algorithm for linguistic analysis. *Journal of Logic and Computation*, 4:1–19.

Lin, Dekang. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

Magerman, David M. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Department of Computer Science, Stanford University.

Marcus, M., G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate–argument structure. In *Proceedings of the Human Language Technology Workshop*, pages 114–119, Princeton, NJ.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

McConnell-Ginet, Sally. 1982. Adverbs and logical form. *Language*, 58:144–184.

McConville, Mark. 2007. *Inheritance and the Categorial Lexicon*. Ph.D. thesis, University of Edinburgh.

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

Mel'čuk, Igor and Nicolaj Pertsov. 1987. *Surface Syntax of English*. John Benjamins, Amsterdam.

Merlo, Paola and Gabriele Musillo. 2005. Accurate function parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 620–627, Vancouver, Canada.

Miyao, Yusuke, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 684–693, Hainan Island, China.

Miyao, Yusuke and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 83–90, Ann Arbor, MI.

Moortgat, Michael. 1988. *Categorial Investigations*. Ph.D. thesis, Universiteit van Amsterdam. Published by Foris, Dordrecht, 1989.

Moortgat, Michael. 1997. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*. North Holland, Amsterdam, pages 93–177.

Moot, Richard. 2002. Parsing corpus-induced type-logical grammars. In *Proceedings of the CoLogNet/ElsNet Workshop on Linguistic Corpora and Logic Based Grammar Formalisms*, pages 70–85, Utrecht, Netherlands.

Morrill, Glyn. 1994. *Type-Logical Grammar*. Kluwer, Dordrecht.

Nunberg, Geoffrey. 1990. *The Linguistics of Punctuation*. Number 18 in CSLI Lecture Notes. CSLI Publications, Stanford, CA.

O'Donovan, Ruth, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III Treebanks. *Computational Linguistics*, 31(3):329–365.

Oflazer, Kemal, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In Anne Abeillé, editor, *Treebanks. Building and using syntactically annotated corpora.* Kluwer Academic Publishers, Amsterdam, pages 261–277.

Osborne, Miles and Ted Briscoe. 1998. Learning Stochastic Categorial Grammars. In *Proceedings of CoNLL97: Computational Natural Language Learning*, pages 80–87, Somerset, NJ.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Pollard, Carl and Ivan Sag. 1992. Anaphors in English and the scope of binding theory. *Linguistic Inquiry*, 23:261–303.

Pollard, Carl and Ivan Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago, IL.

Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA.

Ratnaparkhi, Adwait. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, Computer and Information Science, University of Pennsylvania.

Rooth, Mats and Barbara Partee. 1982. Conjunction, type-ambiguity, and wide-scope 'or'. In *Proceedings of the First West Coast Conference on Formal Linguistics*, pages 353–362, Stanford CA.

Ross, John Robert. 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, MIT. Published as "Infinite Syntax!", Ablex, Norton, NJ. 1986.

Shen, Libin and Aravind Joshi. 2005. Building an LTAG Treebank. Technical Report MS-CIS-05-15, CIS, University of Pennsylvania, Philadelphia, PA.

Stabler, Edward. 2004. Varieties of crossing dependencies: Structure-dependence and mild context sensitivity. *Cognitive Science*, 28:699–720.

Steedman, Mark. 1985. Dependency and coordination in the grammar of Dutch and English. *Language*, 61:523–568.

Steedman, Mark. 1989. Constituency and coordination in a Combinatory grammar. In Mark Baltin and Anthony Kroch, editors, *New Conceptions of Phrase Structure*. Chicago University Press, Chicago IL, pages 201–306.

Steedman, Mark. 1996. *Surface Structure and Interpretation*. MIT Press, Cambridge, MA.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Steedman, Mark and Jason Baldridge. 2006. Combinatory Categorial Grammar. In Keith Brown, editor, *Encyclopedia of Language and Linguistics*, volume 2. Elsevier, Oxford, 2nd edition, pages 610–622.

Trechsel, Frank. 2000. A CCG approach to Tzotzil pied-piping. *Natural Language and Linguistic Theory*, 18:611–663.

Villavicencio, Aline. 2002. *The Acquisition of a Unification-Based Generalised Categorial Grammar*. Ph.D. thesis, Computer Laboratory, University of Cambridge.

White, Michael. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language and Computation*, 4(1):39–75.

White, Michael and Jason Baldridge. 2003. Adapting Chart Realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126, Budapest, Hungary.

Wittenburg, Kent and Robert Wall. 1991. Parsing with categorial grammar in predictive normal form. In Masaru Tomita, editor, *Current Issues in Parsing Technology*. Kluwer, Dordrecht, pages 65–83. Revised selected papers from International Workshop on Parsing Technology (IWPT) 1989, Carnegie Mellon University.

Xia, Fei. 1999. Extracting Tree Adjoining Grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398–403, Beijing, China.

Xia, Fei. 2001. *Automatic Grammar Generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania.

Xia, Fei, Martha Palmer, and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *Proceedings of the 2000 Conference on Empirical Methods in Natural Language Processing*, pages 53–62, Hong Kong.

XTAG-group. 1999. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-98-18, University of Pennsylvania.