Chapter 10

Learning an Italian Categorial Grammar

R. Bernardi, A. Bolognesi, C. Seidenari, F. Tamburini

1. Grammar Learning

Categorial Grammar (CG) is a lexicalized formal grammar well known for its tied connection between syntax and semantics. Variants of it (Combinatory Categorial Grammar, CCG, and Categorial Type Logic, CTL) have been used to reach wide coverage grammars for English (Hockenmaier 2003) and Dutch (Moortgat and Moot 2002). The former has resulted into a large CCG Bank that has been enriched with semantic information (Bos 2005; Clark and Curran 2007; Curran, Clark and Bos 2007). Therefore, CG elegant syntax-semantics interface has already provided promising preliminary results. This connection is even more tied in the CTL framework where it is represented by a formal correspondence between derivations and lambda-calculus rules (viz. Curry-Howard Correspondence (Van Benthem 1986)). In this work we adopted the CTL version of CG. Differently from CCG, composed only by logical rules, CTL is based on logical rules, that create linguistic structures, and structural rules, that take care of cross-linguistic word-order variations.

Following Hockenmaier 2003, the task of learning CTL can be divided into several sub-tasks: (i) learning the types from existing treebanks; (ii) parsing raw corpora to build a CGBank, a bank of derivations; (iii) learning semantic labeling of the derivations. Furthermore, the type learning could be further

enhanced by inducing structural rules that will help filtering out the sets of types without loss of information. In Bernardi and Bolognesi 2006 we have presented a statistical parser to help building a bank of Italian CG derivations. In this paper, we focus on discussing the treebank we start from, the pre-processing work we had to carry out, and presenting our preliminary results.

Our ultimate goal will be the annotation of CORIS/CODIS, a 100-million-word synchronic corpus of contemporary written Italian. Our starting point, instead, is TUT (Turin University Treebank), a collection of syntactically annotated Italian sentences (1,800 sentences) with dependency relations.

This paper has the following structure. In Section 2 we recall grammar formalisms we dealt with in order to obtain a CG treebank. In Section 3 we discuss the preprocessing needed for translating TUT structures into CG binary trees. In Section 4 we study the translation from TUT to CG trees. In Section 4.3 and 5 we briefly discuss future steps we are planning in order to improve our CG treebank. In Section 6 we draw some conclusions.

2. Formal Grammars

Since our starting point is TUT, a dependency treebank, and our goal is to build CG derivations, a first important step is to translate the TUT dependency tree into the latter. Before going into the details of the pre-processing phase, we briefly introduce the two formalisms and highlight their similarity and differences.

2.1. Dependency Grammar and TUT format

The Turin University Treebank (TUT) is a corpus of Italian sentences annotated by specifying relational structures augmented with morpho-syntactic information and semantic role (henceforth ARS) in a monostratal dependency-based representation. The treebank includes 38,653 words and 1,800 sentences from the Italian civil law code, the national newspapers *La Stampa* and *La Repubblica*, and from various reviews, newspapers, novels, and academic papers.

The ARS schema consists of i) morpho-syntactic, ii) functional-syntactic and iii) semantic components, specifying part-of-speech, grammatical relations,

and thematic role information, respectively. The reader is referred to Bosco 2003 for a detailed description of the TUT annotation schema.

Because we are interested in extracting dependency relations, we can focus on the functional-syntactic component of the TUT annotation, where information relating to grammatical relations (heads and dependents) is encoded. In TUT structures, each node is labelled by a word; each edge is labelled by a grammatical relation. The information concerning a single node word is as follows

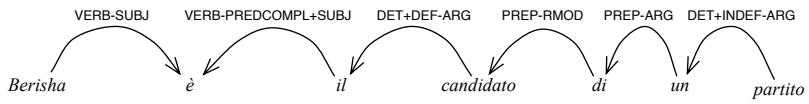
$$n \text{ word } (f_1\ f_2\ \ldots\ f_n)\ [H;MORPH-SYNT-SEM]$$

where, n is the number of the linear order of the word occurrence; f_i are morphological features associated with the word itself; $MORPH-SYNT-SEM$ are the grammatical relation concerning the dependency edge linking the word with its syntactic head (H).

An example is given below (tr. "Berisha is the candidate of a party"): the node TOP-VERB is the root of the whole structure[1].

```
1   Berisha     (Berisha NOUN PROPER)                          [2;VERB-SUBJ]
2   è           (ESSERE VERB MAIN IND PRES INTRANS 3 SING)     [0;TOP-VERB]
3   il          (IL ART DEF M SING)                            [2;VERB-PREDCOMPL+SUBJ]
4   candidato   (CANDIDATO NOUN COMMON M SING)                 [3;DET+DEF-ARG]
5   di          (DI PREP MONO)                                 [4;PREP-RMOD]
6   un          (UN ART INDEF M SING)                          [5;PREP-ARG]
7   partito     (PARTITO NOUN COMMON M SING)                   [6;DET+INDEF-ARG]
8   .           (#.  PUNCT)                                    [2;END]
```

In the following we will use dependency structure format that are easier to read and compare with the CG binary trees: arrows link a dependent with its head by pointing to it and carrying the grammatical relation as illustrated by our running example:



[1]The top nodes used in TUT are TOP-VERB, TOP-NOUN, TOP-CONJ, TOP-ART, TOP-NUM, TOP-PRON, TOP-PHRAS and TOP-PREP.

2.2. Categorial Grammar

Categorial Type Logic (CTL) (Moortgat 1997) is a logic-based formalism belonging to the family of Categorial Grammars (CG). In CTL, the type-forming operations of CG are viewed as logical connectives. As the slogan "Parsing-as-Deduction" suggests, such a view makes it possible to do away with combinatory syntactic rules altogether; establishing the well-formedness of an expression becomes a process of deduction in the logic of the type-forming connectives.

In this framework, The basic distinction is not among head and dependents, but rather between complete and incomplete expressions. Complete expressions are categorized by means of *atomic* type formulas; grammaticality judgments for expressions with an atomic type do not require further contextual information. Typical examples of atomic types would be 'sentence' (S) and 'common noun' (N). Incomplete expressions are categorized by means of fractional type formulas; the denominators of these fractions indicate the material that has to be found in the context in order to obtain a complete expression of the type of the numerator.

10.0.1 Definition[Fractional type formulas]
Given a set of basic types ATOM, the set of types TYPE is the smallest set such that:

1. if $A \in$ ATOM, then $A \in$ TYPE;
2. if A and $B \in$ TYPE, then A/B and $B\backslash A \in$ TYPE.

where $A\backslash B$ (A/B) would be assigned to a structure of category B missing an A on its left (resp. right).

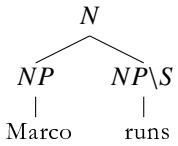
For instance, intransitive verbs as well as verb phrases are assigned the category $NP\backslash S$.

Notice that the language of fractional types is essentially higher-order: the denominator of a fraction does not have to be atomic, but can itself be a fraction. Differently both from classical CG and CCG, the logic family of these grammar formalisms, CTL, besides the logical rules corresponding to function application has those corresponding to abstraction. The latter are

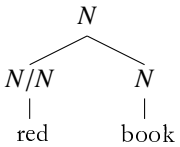
indispensable if one is interested in capturing the full set of theorems of the type calculus. Classical CG (in the style of Ajdukiewicz and Bar-Hillel) uses only the Elimination rules, and hence has restricted inferential capacities. It is impossible in classical CG to obtain the validity $A \vdash B/(A\backslash B)$, for example. We aim to use the full inferential power of the system to reduce the number of category assignments. Still, the classical CG perspective will be useful to realize our aim of automatically learning type assignments from structured data obtained from the TUT corpus thanks to the type resolution algorithm explained in Section 4.

Since we are interested in translating TUT dependency trees into CG binary trees an important aspect to emphasise is the role of head and dependent, argument and modifiers in CG. As in Lexicalised Tree-Adjoining Grammar (LTAG), dependencies are expressed locally within the syntactic type. We illustrate these points by looking at some examples.

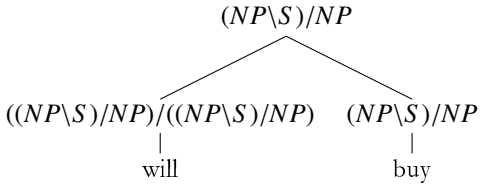
Head vs. Dependent "Marco runs"



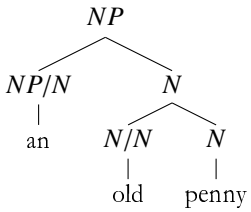
Argument vs. Modifiers "red book"



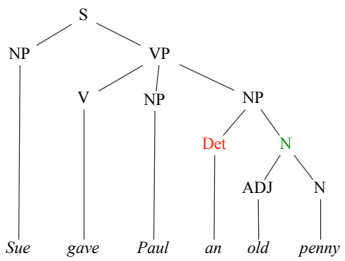
In case of auxiliary verbs, e.g. "will" combined with an untensed verb as e.g. "buy", the dependency of the subject *np* is percolated up from the untensed verb via the auxiliary, and the latter is the head of the phrase:



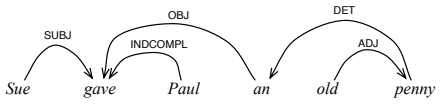
Let us give another example where Head/Dependent and Argument/Modifiers occur together by considering the noun phrase "an old penny".



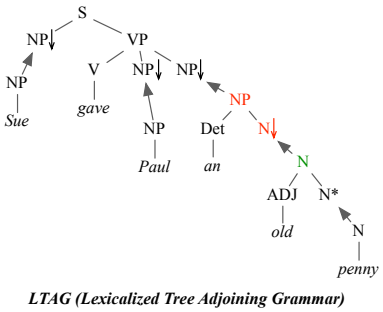
Finally, the difference among constituent, dependency and CG binary trees are illustrated by the example below representing, in different formats, the sentence "Sue gave Paul an old penny".



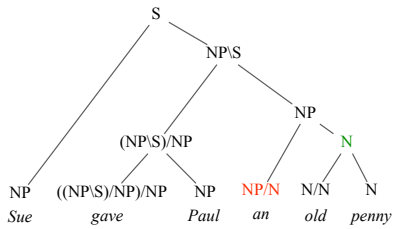
CFG (Context Free Grammar)



DG (Dependency Grammar)



LTAG (Lexicalized Tree Adjoining Grammar)



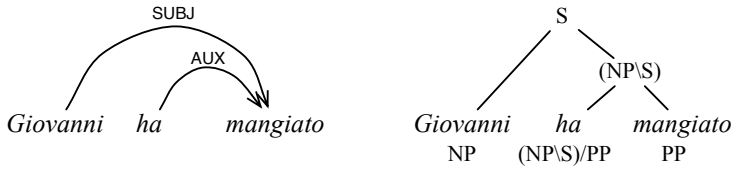
CG (Categorial Grammar)

3. Pre-processing

At this stage there are only three types of dependency-like structures that need to be pre-processed in order to fit our categorial perspective: auxiliar, coordination and relative clause.

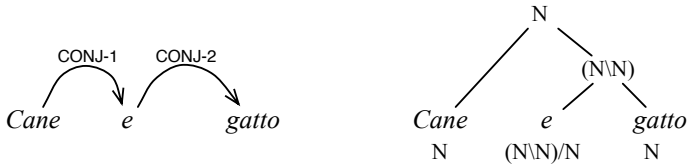
In the TUT treebank, auxiliaries are represented as Dependent on the main verb: in our perspective they should be treated instead as the main Functor taking the participle as the Argument.

The example below shows our perspective for the auxiliary on the right for the sentence "Giovanni ha mangiato" (tr. "Giovanni ate"), where the auxiliary "ha" takes a past participle (PP) on its right and returns a verb phrase (NP\S) looking for a subject ("Giovanni").



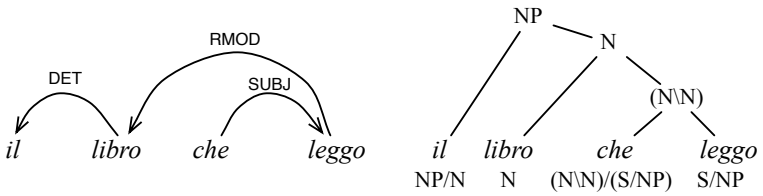
For coordination TUT has chosen what is described as an "asymmetric option", i.e. a representation where the first conjunct is taken as the Head of the coordinator which in turn is taken as the Head of the second conjunct. From our point of view the coordinator should be seen instead as the main Functor, taking the first and the second conjunct as its Arguments.

The example below shows our perspective for the coordinator on the right for the noun phrase "Cane e gatto" (tr. "Dog and cat"), where the coordinator "e" takes the noun "gatto" (N) on its right, then the noun "Cane" (N) on its left and returns a noun.



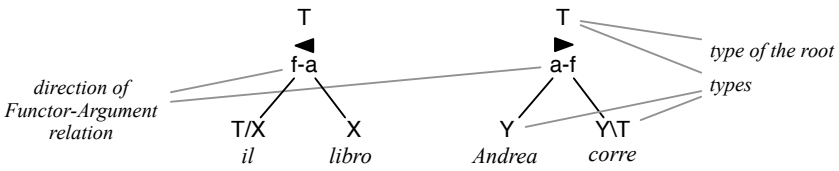
The approach of TUT to the representation of relative clauses implies that 1) the relative pronoun depends on the verb as a standard Argument 2) the verb is the Head of the relative clause and 3) in turn, is connected to the governing noun in the main clause as a Modifier. Our own approach is to select 1) the relative pronoun as the main Functor taking as its Arguments 2) the verb of the relative clause and 3) the noun in the main clause.

The example below shows our perspective for the relative clause inside the noun phrase "il libro che leggo" (tr. "the book I read"), where the relative pronoun "che" takes the verb phrase "leggo" (S/NP) on its right, then the noun "libro" (N) on its left and returns a noun. Note that on the TUT dependency structure on the left the relative pronoun is a dependent of relative verb that has the crucial role of modifying the antecedent in the main phrase.



4. CTL Grammar Learning

Our work is based on the type inference algorithms for CG studied in Buszkowski and Penn 1990 and Buszkowski 1991. The structured data needed by their type inference algorithms are so-called *functor-argument structures* (*fa*-structures). An *fa*-structure for an expression is a binary branching tree; the leaf nodes are labeled by lexical expressions (words), the internal nodes by one of the symbols ◄ (for structures with the functor as the left daughter) or ► (for structures with the functor as the right daughter). An example of fa-structures and of type assignments for them is given below:



To assign types to the leaf nodes of an *fa*-structure, one proceeds in a top-down fashion. The type of the root of the structure is fixed (for example: S). Compound structures are typed as follows:

- to type a structure $\Gamma \blacktriangleleft \Delta$ as A, type Γ as A/B and Δ as B;
- to type a structure $\Gamma \blacktriangleright \Delta$ as A, type Γ as B and Δ as $B\backslash A$.

If a word occurs in different structural environments, the typing algorithm will produce distinct types. The set of type assignments to a word can be reduced by *factoring*: one identifies type assignments that can be unified. For an example, compare the structured input below:

a. Claudia ► parla
b. Claudia ► (parla ► bene)

Assuming a goal type S, from (a) we obtain the assignments

$$\text{Claudia} : A, \text{parla} : A\backslash S$$

and from (b)

$$\text{Claudia} : C, \text{parla} : B, \text{bene} : B\backslash(C\backslash S)$$

Factoring leads to the identifications $A = C$, $B = (A\backslash S)$, producing for "bene" the modifier type $(A\backslash S)\backslash(A\backslash S)$.

Starting from this algorithm our global workplan proceeds as illustrated in Figure 10.1 and detailed in the remaining of this section.

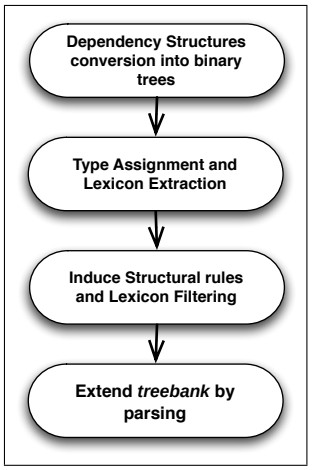
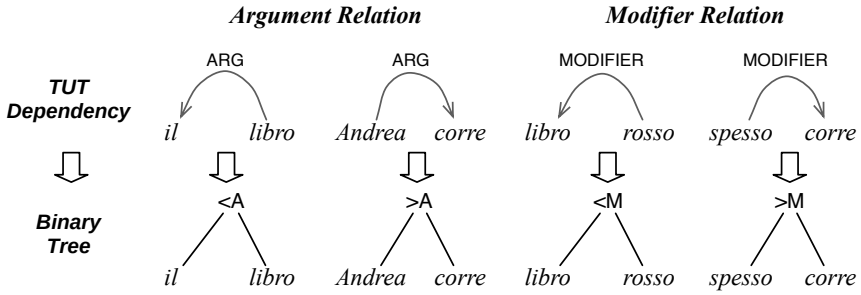


Figure 10.1: Workplan.

4.1. Dependency Structure conversion into binary trees

The first step, consist in the conversion of Dependency Structures into binary trees. The structured data needed for obtaining CG derivations are functor-argument structures.

Our CTL grammar extraction algorithm for the TUT treebank is parametrized in a number of ways: in order to obtain categorial grammar binary tree out of Dependency Structures we focus our attention on SYNT tag as emphasised above. We convert TUT annotated sentences into binary trees on the basis of Head-Dependent relations between lexical entries, and we translate each grammatical relation into the correspondent functor symbols as illustrated below (note that the general f-a symbols ►◄ are replaced by four more descriptive symbols that will lead to a slightly different type assignment method).



For instance, our running example of Section 2.1 is transformed as shown in Figure 10.2.

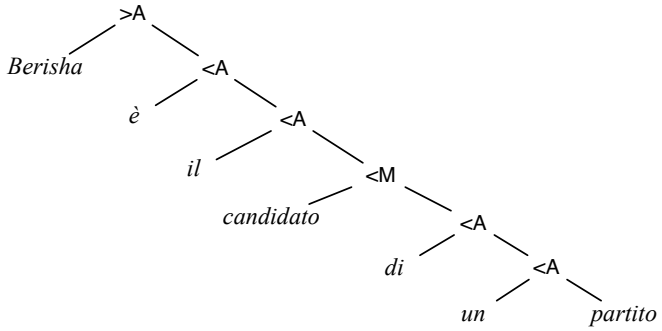
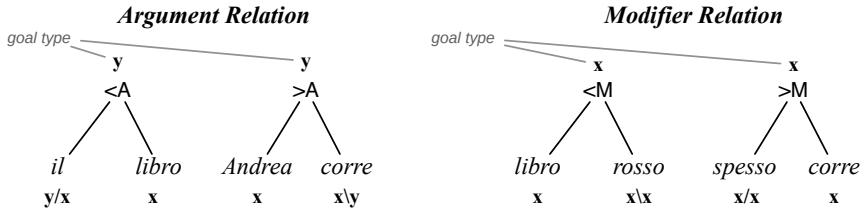


Figure 10.2: Conversion for the sentence example "Berisha è il candidato di un partito".

4.2. Type Assignment and Lexicon Extraction

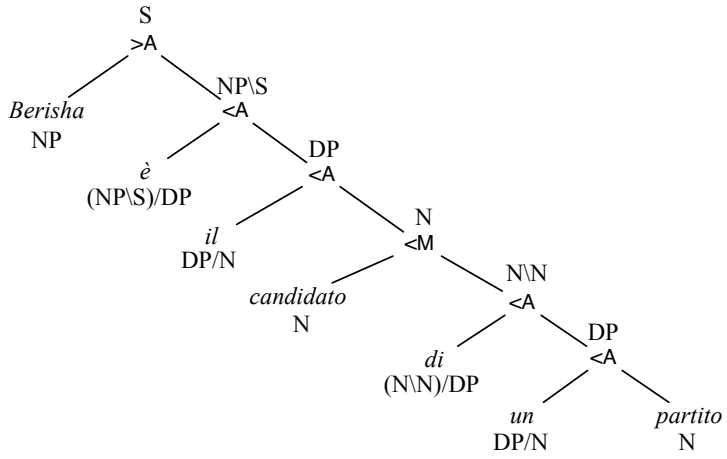
We instantiate atomic categories using the grammatical relations and the PoS information given in TUT. By running the unification algorithm we build a lexicon containing all the types obtained per each word. The Type Assignment procedure can be summarized in two steps:

- apply the type assignment algorithm (Buszkowski and Penn 1990) to the obtained binary trees, according to the following rules:



- set atomic categories on the basis of grammatical relations,
  - focusing on `SYNT` tag, and
  - on `PoS` information

An example of type assignment for the running example of Section 2.1 is given below:



4.3. Structural Rules Induction and Lexicon Filtering

In this section we briefly describe the step of 'structural rules induction and lexicon filtering' we are currently working on, that corresponds to step 3 as indicated in the workflow of Figure 10.1. Structural rules (Moortgat 1997; Moortgat and Moot 2002; Moortgat 2001) are special rules we can add to the

logical framework in order to minimize the lexical ambiguity and so reduce the number of types assigned to each word. In order to induce structural rules from our treebank we need information on the mode of composition, that is labels which describe the grammatical relation under the slashes. These labels are taken from the labels on the edges of TUT dependency structures. Hence, those words that receive too many lexicon assignment can be filtered by structural rules.

5. Treebank Extension

The next step following our workflow in Figure 10.1 consists in using the statistical parser we proposed in Bernardi and Bolognesi 2006 in order to extend the treebank.

To run a first experiment, we chose to start from a subset of TUT that contains dependency structures with a low level of structural complexity. To this end, we have adopted the structural complexity definition proposed in Lin 1996: the structural complexity of a dependency structure is the total length of the dependency links in the structure, where the length of a dependency link is one plus the number of words between the head and the dependent. This made possible a first grammar learning starting from a dependency bank with simple sentences.

From the 1800 sentences of TUT we extracted 443 dependency structures with structural complexity less than 70, obtaining our initial gold standard. Then we translated these trees into a CTL derivations bank as explained in Section 4.

So far, we have extracted statistical information only for the first 400 trees, leading to the creation of the training set of trees. The remaining 43 trees formed the test set. The lexicon obtained consists of 1909 words, 480 categories, with an average of two categories per word.

Refer to Bernardi and Bolognesi 2006 for a complete description of the experiments and an in-depth evaluation of parser performances.

6. Conclusions and Future Work

We described the preliminary phases necessary to learn a CGBank, namely the pre-processing operation, the conversion of dependency structures into binary trees, and the extraction of lexicon type assignments. Furthermore, we have described the next steps we will need to work on, namely inducing structural rules and filtering lexicon entries. The last steps of the work will require the conversion of the binary trees into a CTL derivation bank and the extension of it by means of parsing and evaluating new raw texts. To this end we have developed and trained a statistical parser (Bernardi and Bolognesi 2006).

We are currently improving our learning of Type Assignment and Structural Rules. Then, we will transform the binary trees obtained with their assigned types into an actual CTL derivations bank by exploiting the Structural Rules we have induced.

Furthermore, we are planning to extend the CTL derivations bank by extending the original treebank applying the same grammar learning method to VIT (Venice Italian Treebank), a collection of syntactically annotated Italian spoken and written sentences (300.000 words) (Delmonte 2004).

References

Bernardi, R. and Bolognesi, A. (2006). "Building an italian CG bank via incremental statistical parsing". In *Proc. of Fifth Workshop on Treebanks and Linguistic Theories*, Ufal, pp. 223–234.

Bos, J. (2005). "Towards wide-coverage semantic interpretation". In *Proc. of Sixth International Workshop on Computational Semantics IWCS-6*, Tilburg, pp. 42–53.

Bosco, C. (2003). *A grammatical relation system for treebank annotation*. Ph.D. Thesis, Computer Science Department, Turin University.

Buszkowski, W. (1991). "On Generative Capacity of the Lambek Calculus". In *Proc. of European Workshop on Logics in AI*, pp. 139–152.

Buszkowski, W. and Penn, G. (1990). "Categorial grammars determined from linguistic data by unification". *Studia Logica*, 49, pp. 431–454.

Clark, S. and Curran, J. R. (2007). "Formalism-Independent Parser Evaluation with CCG and DepBank". In *Proc. of 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 248–255.

Curran, J. R., Clark, S. and Bos, J. (2007). "Linguistically Motivated Large-Scale NLP with C&C and Boxer". In *Proc. of ACL 2007 Demonstrations (ACL demo)*, pp. 33–36.

Delmonte, R. (2004). "Strutture sintattiche dall'analisi computazionale di corpora di italiano". In Cardinaletti, A. and Frasnedi, F. (Eds.), *Intorno all'italiano contemporaneo. Tra linguistica e didattica*, Milano: F. Angeli, pp. 187–220.

Hockenmaier, J. (2003). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. Thesis, School of Informatics, University of Edinburgh.

Lin, D. (1996). "On the structural complexity of natural language sentences". In *Proc. of 16th conference on Computational linguistics*, Morristown, Association for Computational Linguistics, pp. 729–733.

Moortgat, M. (1997). "Categorial type logics". In Van Benthem, J. and Ter Meulen, A. (Eds.), *Handbook of Logic and Language*, Cambridge, MA: MIT Press, pp. 93–178.

Moortgat, M. (2001). "Structural equations in language learning". In De Groote, P., Morrill, G. and Retoré, C. (Eds.), *Logical Aspects of Computational Linguistics*, Berlin: Springer, pp. 1–16.

Moortgat, M. and Moot, R. (2002). "Using the spoken dutch corpus for type-logical grammar induction". In *Proc. of Third International Language Resources and Evaluation Conference*, Las Palmas—Canary Islands, pp. 419–425.

Van Benthem, J. (1986). *Essays in logical semantics*. Dordrecht: Reidel Publishing Company.