A Tableau Prover for Natural Logic and Language

Lasha Abzianidze
TiLPS, Tilburg University, the Netherlands
L.Abzianidze@uvt.nl

Abstract

Modeling the entailment relation over sentences is one of the generic problems of natural language understanding. In order to account for this problem, we design a theorem prover for Natural Logic, a logic whose terms resemble natural language expressions. The prover is based on an analytic tableau method and employs syntactically and semantically motivated schematic rules. Pairing the prover with a preprocessor, which generates formulas of Natural Logic from linguistic expressions, results in a proof system for natural language. It is shown that the system obtains a comparable accuracy ($\approx 81\%$) on the unseen SICK data while achieving the state-of-the-art precision ($\approx 98\%$).

1 Introduction

A problem of recognizing textual entailments (RTE)—given two text fragments T (for a text) and H (for a hypothesis), determine whether T entails, contradicts or is neutral to H—is considered as a complex and, at the same time, fundamental problem for several NLP tasks (Dagan et al., 2005). For more than a decade, RTE challenges have been held, where systems are competing to each other with respect to human annotated RTE test data; but there are few systems that try to solve RTE problems by computing meanings of linguistic expressions and employing inference engines similar to proof procedures of formal logics. Moreover, those few systems are usually used in combination with shallow classifiers since the systems' performances alone are poor.

The current paper advocates that purely deductive inference engines over linguistic representations backed up with a simple lexical knowledge base could be solely and successfully used for the RTE task. Our work builds on the theory of an analytic tableau system for Natural Logic (Natural Tableau) introduced by Muskens (2010). The theory offers to employ a tableau method—a proof procedure used for many formal logics—for the version of Natural Logic that employs Lambda Logical Forms (LLFs)—certain terms of simply typed λ-calculus—as Logical Forms (LFs) of linguistic expressions. The merits of the current approach are several and they can be grouped in two categories: virtues attributed to the tableau prover are (i) the high precision for the RTE task characteristic to proof procedures, (ii) the transparency of the reasoning process, and (iii) ability for solving problems with several premises; and those concerning LLFs are (iv) an evidence for LFs that are reminiscent of Surface Forms but still retaining complex semantics, and (v) an automatized way of obtaining LLFs from wide-coverage texts.

The rest of the paper is organized as follows. First, Natural Tableau is introduced, and then a method of obtaining LLFs from raw text is described. We outline the architecture of an implemented theorem prover that is based on the theory of Natural Tableau. The power of the prover is evaluated against the SICK data; the results are analyzed and compared to related RTE systems. The paper concludes with future work.

2 Natural Tableau for Natural Logic

Natural Logic is a vague notion and refers to logics that account for valid inferences of natural languages, where reasoning and the grammar are strongly related to each other and LFs resemble surface forms (Lakoff, 1972). On the other hand, a tableau method (Beth, 1955) is a popular proof procedure and nowadays many formal logics have their own version of it (D'Agostino et al., 1999). A combination of these two devices is offered by Muskens (2010), where the language of Natural Logic is considered to be a part of simply typed

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2492–2502,
Lisbon, Portugal, 17-21 September 2015.

$$\frac{X\ A\ B:\ [\,]:\ \mathbb{F}}{\begin{array}{c}A:\ [c]:\ \mathbb{T}\\ B:\ [c]:\ \mathbb{F}\\ \text{s.t. } X \in \{\texttt{all}, \texttt{every}\}\\ \text{and } c \text{ is a } \textit{fresh} \text{ term}\end{array}}\forall_F \qquad \frac{A\ B:\ [\vec{C}]:\ \mathbb{X}}{A:\ [B,\vec{C}]:\ \mathbb{X}}\text{PUSH} \qquad \frac{A:\ [B,\vec{C}]:\ \mathbb{X}}{A\ B:\ [\vec{C}]:\ \mathbb{X}}\text{PULL}$$

$$\frac{\begin{array}{c}A:\ [\vec{C}]:\ \mathbb{T}\\ B:\ [\vec{C}]:\ \mathbb{F}\end{array}}{\begin{array}{c}\times\\ \text{s.t. } A \leq B\end{array}}\leq\!\times \qquad \frac{X\ A\ B:\ [\,]:\ \mathbb{F}}{\begin{array}{c}A:\ [d]:\ \mathbb{F} \qquad B:\ [d]:\ \mathbb{F}\\ \text{s.t. } X \in \{\texttt{some}, \texttt{a}\} \text{ and } d \text{ is an } \textit{old} \text{ term}\end{array}}\exists_F$$

$$\frac{\texttt{not}\ A:\ [\vec{C}]:\ \mathbb{X}}{A:\ [\vec{C}]:\ \overline{\mathbb{X}}}\text{NOT}$$

Figure 1: Tableau rules for quantifiers ($\forall_F$ and $\exists_F$), Boolean operators (NOT), formatting (PUSH and PULL) and inconsistency ($\leq\!\times$). The relation $\leq$ stands for entailment, $\vec{C}$ and $\mathbb{X}$ are meta-variables over sequences of terms and truth signs ($\mathbb{T}$ and $\mathbb{F}$), respectively; the bar operator $\overline{\mathbb{X}}$ negates a sign.

λ-terms that are built up from variables and lexical constant terms with the help of application and lambda abstraction. The terms of the language are called LLFs and resemble linguistic surface forms:[1]

$$\texttt{a}_{(et)(et)t}\ \texttt{bird}_{et}\ \texttt{fly}_{et}$$

$$\texttt{some}_{(et)(et)t}\ \texttt{bird}_{et}\ (\texttt{not}_{(et)et}\ \texttt{fly}_{et})$$

$$\texttt{not}_{((et)(et)t)(et)(et)t}\ \texttt{all}_{(et)(et)t}\ \texttt{bird}_{et}\ \texttt{fly}_{et}$$

Note that common nouns and intransitive verbs are typed as properties (i.e. functions from entities to truth values) and quantifiers as binary relations over properties; the latter typing treats quantified noun phrases (QNPs) as generalized quantifiers (GQs)—a term of type properties over properties $(et)t$.

A Natural Tableau entry is a tuple containing a term, a sequence of terms representing an argument list, and a truth sign. The entries are such that when a term is applied to all arguments from an argument list in the order of the list, the resulted term is of type truth value. For example, $A_{eet}c_e : [d_e] : \mathbb{T}$ is a valid tableau entry (i.e. a node) since it consists of a term $A_{eet}c_e$, an argument list $[d_e]$ and a truth sign $\mathbb{T}$ standing for *true*, and additionally, $A_{eet}c_e d_e$ is a term of type t.

A tableau method is a refutation method and it proves an argument by searching a counterexample. The search process is guided by applications of certain set of rules. A tableau rule is a schema with a set of antecedent nodes above a line and a set of precedent branches below a line, where each branch consists of (precedent) nodes. A rule is applicable if all its antecedent nodes match to some nodes in a tableau, and after the rule is applied, precedent nodes of the rule are introduced in the tableau. A tableau consists of branches where each branch models a situation and is either closed (i.e., inconsistent) or open (i.e., consistent) depending whether it contains a closure $\times$ sign (i.e., an obvious contradiction). A tableau is closed if all its branches are closed, otherwise it is open.

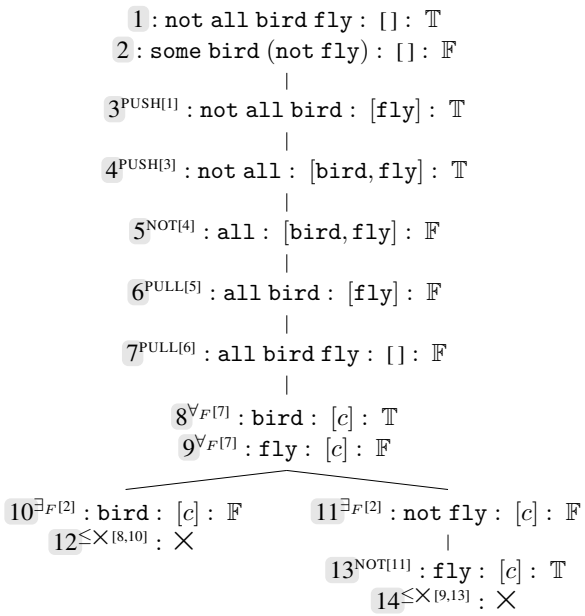

Figure 2: The closed tableau serves as a proof for: *not all birds fly → some bird does not fly*

In Figure 2, a tableau proof, which employs the rules of (Muskens, 2010) from Figure 1, is presented. In order to show a way the tableau is developed, the nodes are enumerated and annotated with a source rule and IDs of nodes from which a current node is obtained. For example, 3 is obtained from 1 by the PUSH rule. In order to prove an argument, the tableau starts with a counterexample of the argument, i.e. a premise being true and a conclusion false. After several rule applications, all the branches of the tableau close meaning

[1]Since modeling intensionality is beyond the scope of the paper, we present LLFs typed with extensional semantic types, i.e. types are built up from basic e (for entities) and t (for truth values) types. We use the comma as a type constructor, e.g., (e, t) stands for a functional type from entities to truth values. The comma is omitted when types are denoted by single letters, e.g., et stands for (e, t). Taking into account right-associativity of the type constructor we often drop parentheses for better readability. Terms are optionally annotated with their types in a subscript.

that none of the situations for the counterexample were consistent.

An advantage of Natural Tableau is that it treats both single and multi-premised arguments in the same fashion and represents a deductive procedure in an intuitive and transparent way.

3 Obtaining LLFs for Natural Tableau

3.1 CCG and the C&C Parser

Combinatory Categorial Grammar (CCG) is a lexicalized grammar formalism that assigns a syntactic category and a semantic interpretation to lexical items, where the items are combined via combinatory rules (Steedman, 2000; Steedman and Baldridge, 2011). The CCG category A/B (or $A\backslash B$) is a category of an item that becomes of category A when it is combined with an item of category B on its right (or left, respectively) side. In the example below, the sentence *every man walks* is analyzed in the CCG formalism, where lexical items are combined via the *forward application* rule and unspecified semantic interpretations are written in a boldface:

$$\dfrac{\dfrac{\dfrac{\text{every}}{(S/(S\backslash NP))/N : \mathbf{every}} \quad \dfrac{\text{man}}{N : \mathbf{man}}}{S/(S\backslash NP) : \mathbf{every\ man}} \quad \dfrac{\text{walks}}{S\backslash NP : \mathbf{walk}}}{S : (\mathbf{every\ man})\ \mathbf{walk}}$$

The CCG derivation trees are suitable structures for obtaining LLFs for at least two reasons. First, the CCG framework is characterized by a transparent interface between syntactic categories and semantic types; second, there exist efficient and robust CCG parsers for wide-coverage texts.

During obtaining LLFs, we employ the C&C CCG parser of Clark and Curran (2007) and EasyCCG of Lewis and Steedman (2014). While the C&C parser is a pipeline of several NLP systems: POS-tagger, chunker, named entity recognizer (NER), lemmatizer (Minnen et al., 2001) supertagger and sub-parser, EasyCCG is an extremely simple but still comparably accurate CCG parser based on A* parsing.[2] These two parsers use different settings for supertagging and parsing; therefore, it is interesting to test both parsers for our application.

In Figure 3, there is a CCG derivation by the C&C parser displayed in a tree style: terminal nodes are annotated with tokens, syntactic categories, lemmas and POS-tags while non-terminal nodes are marked with combinatory rules and resulted categories; some basic categories are subcategorized by features.

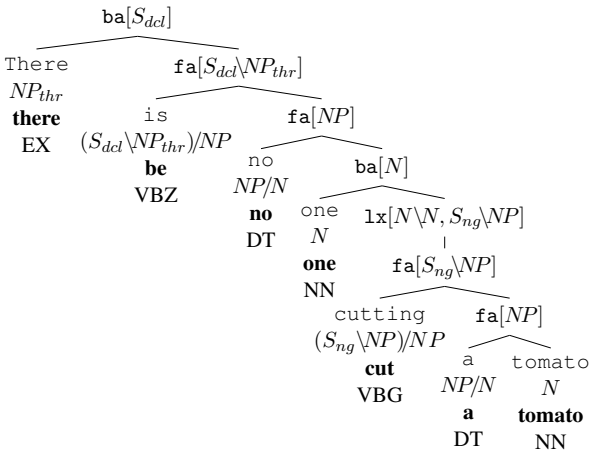

Figure 3: The CCG tree by the C&C parser for *there is no one cutting a tomato* (SICK-2404), where *thr, dcl, ng* category features stand for an expletive *there*, declarative and present participle, respectively.

3.2 From CCG Trees to LLFs

Initially, it may seem easy to obtain fine-grained LLFs from CCG trees of the parsers, but careful observation on the trees reveals several complications. The transparency between the categories and types is violated by the parsers as they employ lexical (i.e. type-changing) rules—combinatory rules, *non-native* ones for CCG, which changes categories. Lexical rules were initially introduced in CCGbank (Hockenmaier and Steedman, 2007) to decrease the total number of categories and rules. In the tree of Figure 3, a lexical rule changes a category $S_{ng}\backslash NP$ of a phrase *cutting a tomato* with $N\backslash N$. In addition to this problem, the trees contain mistakes from supertaggers (and from the other tools, in case of the C&C parser).

The first step in processing CCG trees is to remove directionality from the categories. This step is the same as obtaining unspecified semantic interpretation of a phrase in the CCG framework. While converting categories $A\backslash B$ and A/B into a non-directional type $(\mathtt{b}, \mathtt{a})$, the arrangement of nodes must be changed in a corresponding way. For instance, in case of the top backward application rule ($\mathtt{ba}[S_{dcl}]$ in Figure 3), the order of

[2] The employed C&C parser is trained on *rebanked* CCGbank (Honnibal et al., 2010)—an updated version of CCGbank (Hockenmaier and Steedman, 2007) with improved analyses for predicate-argument structures and nominal modifiers. For EasyCCG, input sentences are already processed by the POS-tagger and the NER of the C&C parser.

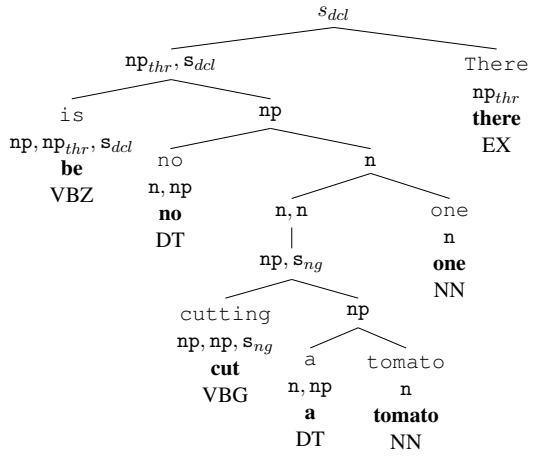

Figure 4: A CCG term obtained from the CCG tree of Figure 3. Categories are converted into types.

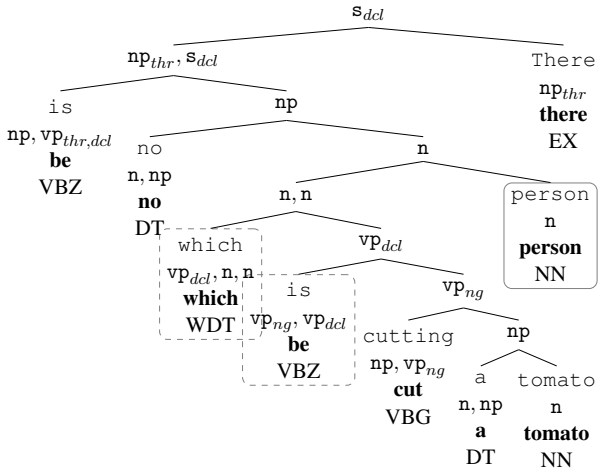

Figure 5: A fixed CCG term that is obtained from the CCG term of Figure 4. A node with a dashed (solid) frame is inserted (substituted, respectively). A type $\mathtt{vp}_{a,b}$ abbreviates $(\mathtt{np}_a, \mathtt{s}_b)$.

nodes is reversed to guarantee that the function category $(\mathtt{s}_{dcl}, \mathtt{np}_{thr})$ precedes its argument category $\mathtt{np}_{thr}$. There are about 20 combinatory rules used by the parsers and for each of them we design a way of reordering subtrees. In the end, the order of nodes coincides with the order according to which semantic interpretations are combined. The reordering recipes for each combinatory rule is quite intuitive and can be found in (Steedman, 2000) and (Bos, 2009), where the latter work also uses the C&C parser to obtain semantic interpretations. Trees obtained after removing the directionality from the categories are called CCG terms since they resemble syntactic trees of typed λ-terms (see Figure 4).

$$\mathtt{on}_{\mathtt{np},\mathtt{pp}}(\mathtt{ice}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{on}_{\mathtt{np},\mathtt{pp}}(\mathtt{a}_{\mathtt{n},\mathtt{np}}\mathtt{ice}_{\mathtt{n}}) \quad (1)$$

$$\mathtt{run}_{\mathtt{np},\mathtt{s}}(\mathtt{dogs}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{run}_{\mathtt{np},\mathtt{s}}(\mathtt{s}_{\mathtt{n},\mathtt{np}}\mathtt{dog}_{\mathtt{n}}) \quad (2)$$

$$(\mathtt{Dow}^{\mathtt{PER}}_{\mathtt{n},\mathtt{n}}\ \mathtt{Jones}^{\mathtt{PER}}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{Dow\_Jones}_{\mathtt{np}} \quad (3)$$

$$(\mathtt{two}_{\mathtt{n},\mathtt{n}}\ \mathtt{dogs}_{\mathtt{n}})_{\mathtt{np}} \longrightarrow \mathtt{two}_{\mathtt{n},\mathtt{np}}\ \mathtt{dogs}_{\mathtt{n}} \quad (4)$$

$$\mathtt{her}_{(\mathtt{pp},\mathtt{n}),\mathtt{np}}\ \mathtt{car}_{\mathtt{pp},\mathtt{n}} \longrightarrow \mathtt{her}_{\mathtt{n},\mathtt{np}}\ \mathtt{car}_{\mathtt{n}} \quad (5)$$

$$\mathtt{who}_{\mathtt{w}}V(Q_{\mathtt{n},\mathtt{np}}N) \longrightarrow Q_{\mathtt{n},\mathtt{np}}(\mathtt{who}_{\mathtt{w}'}VN) \quad (6)$$

$$\mathtt{nobody} \longrightarrow \mathtt{no}_{\mathtt{n},\mathtt{np}}\ \mathtt{person}_{\mathtt{n}} \quad (7)$$

Lexical rules are the third most commonly used combinatory rules (7% of all rules) by the parsers on the SICK data (Marelli et al., 2014b), and therefore, they deserve special attention. In order to compositionally *explain* several category changes made by lexical rules (represented with $(.)_\alpha$ operator in terms), either types of constant terms are set to proper types or lexical entries are inserted in CCG terms. For explaining a lexical rule $\mathtt{n} \rightsquigarrow \mathtt{np}$, mainly used for bare nouns, an indefinite determiner is inserted for singular nouns (1) and a plural morpheme *s* is used as a quantifier for plurals (2). Also identifying proper names with the feature assigned by the C&C NER tool helps to eliminate $\mathtt{n} \rightsquigarrow \mathtt{np}$ change (3). Correcting the type of a quantifier that is treated as a noun modifier is another way of eliminating this lexical rule (4). In case of $(\mathtt{s}, \mathtt{np}) \rightsquigarrow (\mathtt{n}, \mathtt{n})$ change, *which is* phrase is inserted and salvages the category-type transparency of CCG (see Figure 5). As a whole, the designed procedures explain around 99% of lexical rules used in CCG terms of the SICK sentences. Note that explaining lexical rules guarantees a well-formed CCG term in the end.

Apart from the elimination of lexical rules, we also manually design several procedures that fix a CCG term: make it more semantically adequate or simplify it. For example, the C&C parser assigns a category *N/PP* of relational nouns to nouns that are preceded by possessives. In these cases, a type $\mathtt{n}$ is assigned to a noun and a type of possessive is changed accordingly (5). To make a term semantically more adequate, a relative clause is attached to a noun instead of a noun phrase (6), where a type $\mathtt{w} \equiv (\mathtt{vp}, \mathtt{np}, \mathtt{s})$ of a wh-word is changed with $\mathtt{w}' \equiv (\mathtt{vp}, \mathtt{n}, \mathtt{s})$. CCG terms are simplified by substituting terms for *no one, nobody, everyone*, etc. with their synonymous terms (see (7) and Figure 5). These substitutions decrease a vocabulary size, and hence, decrease the number of tableau rules.

The final operation is to convert a fixed CCG term into an LLF, meaning to convert QNPs into GQs of (Montague, 1974; Barwise and Cooper, 1981). In this procedure, a type $(\mathtt{n}, \mathtt{np})$ of a quan-

tifier is replaced with $(\mathtt{n},(\mathtt{np},\mathtt{s}),\mathtt{s})$, and the resulted new NP is applied to the smallest clause it occurs in; but if there are other QNPs too, then it also applies to the clauses where other QNPs are situated. This operation is not deterministic and can return several terms due to multi-options in quantifier scope ordering. As an example, two λ-terms, (9) and (10), are obtained from the CCG term (8) of Figure 5.[3]

$$\mathbf{b}(\mathbf{no}(\mathbf{w}(\mathbf{b}(\mathbf{c}(\mathbf{a}\,\mathbf{t})))\mathbf{p}))\mathbf{th} \quad (8)$$

$$\mathbf{no}\left(\mathbf{w}(\mathbf{b}(\lambda x.\,\mathbf{a}\,\mathbf{t}(\lambda y.\,\mathbf{c}yx)))\mathbf{p}\right)\left(\lambda z.\,\mathbf{b}\,z\,\mathbf{th}\right) \quad (9)$$

$$\mathbf{a}\,\mathbf{t}(\lambda x.\,\mathbf{no}\left(\mathbf{w}(\mathbf{b}(\mathbf{c}x))\mathbf{p}\right)\left(\lambda z.\,\mathbf{b}\,z\,\mathbf{th}\right)) \quad (10)$$

Eventually the final λ-terms, analogous to (9) and (10), obtained from CCG trees will be considered as LLFs that will be used in the wide-coverage theorem prover. It has to be stressed that generated LLFs are theory-independent abstract semantic representation. Any work obtaining semantic representations from CCG derivations can combine its lexicon with (already corrected) LLFs and produce more adequate semantics in this way.

3.3 Extending the Type System

An obvious and simple way to integrate the LLFs, obtained in Subsection 3.2, in Natural Tableau is to translate their types into semantic types built up from e and t.[4] We will not do so, because this means the information loss since the information about syntactic types are erased; for example, usually syntactic types $\mathtt{pp}$, $\mathtt{n}$ and $(\mathtt{np},\mathtt{s})$ are translated as et type. Retaining syntactic types also contributes to fine-grained matching of nodes during rule application in the prover. For instance, without syntactic types it is more complex to determine the context in which a term $\mathbf{game}$ occurs and find an appropriate tableau rule when considering the following LLFs, $\mathbf{game}_{\mathtt{n},\mathtt{n}}\mathbf{theory}$ and $\mathbf{game}_{\mathtt{pp},\mathtt{n}}(\mathbf{of}\ X)$, as both $(\mathtt{n},\mathtt{n})$ and $(\mathtt{pp},\mathtt{n})$ are usually translated into $(et)et$ type, like it is done by (Bos, 2009).

In order to accommodate the LLFs with syntactic types in LLFs of (Muskens, 2010), we extend the semantic type system with $\mathtt{np},\mathtt{n},\mathtt{s},\mathtt{pp}$ basic syntactic types corresponding to basic CCG categories. Thus complex types are now built up from the set $\{e,t,\mathtt{np},\mathtt{n},\mathtt{s},\mathtt{pp}\}$ of types. The extension automatically licenses LLFs with syntactic types as terms of the extended language.

We go further and establish interaction between semantic and syntactic types in terms of a subtyping $\sqsubseteq$ relation. The relation is defined as a *partial order* over types and satisfies the following conditions for any $\alpha_1,\alpha_2,\beta_1,$ and β_2 types:

(a) $e\sqsubseteq\mathtt{np},\quad \mathtt{s}\sqsubseteq t,\quad \mathtt{n}\sqsubseteq et,\quad \mathtt{pp}\sqsubseteq et;$

(b) $(\alpha_1,\alpha_2)\sqsubseteq(\beta_1,\beta_2)$ iff $\beta_1\sqsubseteq\alpha_1$ and $\alpha_2\sqsubseteq\beta_2;$

Moreover, we add an additional typing rule to the calculus: a term is of type β if it is already of type α and $\alpha\sqsubseteq\beta$. According to this typing rule, now a term can be of multiple types. For example, both $\mathbf{walk}_{\mathtt{np},\mathtt{s}}$ and $\mathbf{man}_{\mathtt{n}}$ terms are also of type et, and all terms of type $\mathtt{s}$ are of type t too. From this point on we will use a boldface style for lexical constants of syntactic types.

Initially it may seem that the lexicon of constant terms is doubled in size, but this is not the case as several syntactic constants can mirror their semantic counterparts. This is achieved by *multiple typing* which enables to put semantic and syntactic terms in the same term. For instance, $\mathbf{love}_{\mathtt{np},\mathtt{np},\mathtt{s}}\,c_e\,\mathbf{john}_{\mathtt{np}}$ and $\mathbf{at}_{\mathtt{np},\mathtt{pp}}\,c_e\,d_e$ are well-formed LLFs of type t that combine terms of syntactic and semantic types, and there is no need of introducing semantic terms (e.g., $\mathtt{at}_{\mathtt{eet}}$ or $\mathtt{love}_{\mathtt{eet}}$) in order to have a well-formed term. In the end, the extension of the language is conservative in the sense that LLFs and the tableau proof of Section 2 are preserved. The latter is the case since the tableau rules are naturally extensible to match new LLFs.

4 Implementation of the Prover

In order to further develop and evaluate Natural Tableau, we implement the prover, LangPro, based on the extended theory. Its general architecture is based on the first-order logic (FOL) prover of Fitting (1990). The prover also contains a module for λ-calculus that roughly follows (Blackburn and Bos, 2005).

Setup of the inventory of rules is a crucial for efficiency of the prover. There is a priority order for the categories of rules according to their computational efficiency. The prover most prefers to employ non-branching rules that introduce no fresh terms and antecedents of which can be ignored af-

[3] We use initial letters of lemmas to abbreviate a term corresponding to a lexical entry. Note that (9) represents a reading with *no one* having a wide scope while, in (10), *a tomato* has a wide scope.

[4] The similar translation is carried out in (Bos, 2009) for Boxer (Bos, 2008), where basic CCG categories are mapped to semantic types and the mapping is isomorphically extended to complex categories.

ter the application (e.g., NOT). Less preferred and inefficient rules are the ones that branch, produce new terms or antecedents of which are kept after the application (e.g., $\forall_F$ and $\exists_F$). In order to encourage finding short proofs, *admissible* rules representing shortcuts of several rule applications are also introduced (e.g., FUN$\uparrow$ and ARG in Figure 9). The inventory consists of about 50 rules, where most of them are manually designed based on RTE problems (see Section 5.1) and the rest represents the essential part of the rules found in (Muskens, 2010).

The LLF generator (LLFgen) is a procedure that generates LLFs from a CCG derivation in the way described in Subsection 3.2. We also implement an LLF-aligner that serves as an optional preprocessor between LLFgen and the prover itself; it aligns identical chunks of LLFs and treats them as a constant (i.e. having no internal structure). This treatment often leads to smaller tableau proofs. The example of aligned LLFs is given in Figure 8.

LangPro uses only the antonymy relation and a transitive closure of the hyponymy/hypernymy relations from WordNet 3.0 (Fellbaum, 1998) as its knowledge base (KB). The entailment $\leq$ (contradiction $\perp$) relation between lexical constants of the same type $A \leq B$ ($A \perp B$) holds if there exists a WordNet sense of A that is a transitive hyponym (an antonym) of some WordNet sense of B. Note that there is no word sense disambiguation (WSD) used by the prover; therefore, adopting these interpretations of entailment and contradiction amounts to considering all senses of the words. For example, *a man is crying* entails *a man is screaming* as there are senses of *cry* and *scream* that are in the entailment relation.

All in all, chaining a CCG parser, LLFgen, the LLF-aligner, the prover and KB results in an automatized tableau prover LangPro which operates directly over natural language text.

5 Learning and Evaluation

5.1 Learning

For learning and evaluation purposes, we use the SICK data (Marelli et al., 2014b). The data consists of problems that are rich in the lexical, syntactic and semantic phenomena that compositional distributional semantic models (Mitchell and Lapata, 2010) are expected to account for.[5] The SICK data contains around 10K text-hypothesis pairs that are classified in three categories: entailment, contradiction, and neutral.

During learning we used only the trial portion of the data, SICK-trial, including 500 problems. The learning process consists of improving the components of the prover while solving the RTE problems: designing fixing procedures of LLFgen, adding new sound rules to the inventory, and introducing valid relations in KB that were not found in WordNet (e.g., *woman*$\leq$*lady*, *note*$\leq$*paper* and *food*$\leq$*meal*). During learning, each RTE problem is processed as follows:

input: $(T, H, \texttt{answer})$;
1: $t =$ the first LLF of $\texttt{llf}(T)$;
2: $h =$ the first LLF of $\texttt{llf}(H)$;
3: **case** answer, $\texttt{tab}\{t{:}\mathbb{T}, h{:}\mathbb{F}\}$, $\texttt{tab}\{t{:}\mathbb{T}, h{:}\mathbb{T}\}$
ENTAILMENT, CLOSED, OPEN: HALT;
CONTRADICTION, OPEN, CLOSED: HALT;
NEUTRAL, OPEN, OPEN: HALT;
4: **otherwise**
5: **if** t or h is incorrect **then** try to amend llf; go to 1
6: **else if** a rule is missing **then** add it; go to 3
7: **else if** a relation is missing **then** add it; go to 3
8: **else** HALT;

A function llf denotes the combination of LLFgen and a CCG parser; for learning we use only the C&C parser. A function $\texttt{tab} : S \rightarrow \{\text{CLOSED}, \text{OPEN}\}$ returns CLOSED if one of the tableaux initiated with aligned or non-aligned set S of nodes closes; otherwise it returns OPEN. For instance, while checking a problem (T, H) on entailment (contradiction), tableau starts with a counterexample: T being true and H false (true, respectively). Note that 5-7 procedures are carried out manually while the phase is significantly facilitated by graphical proofs produced by LangPro.[6]

As a result, there were collected around 30 new rules where about a third of them are admissible ones; the new rules cover phenomena like noun and adverbial modifiers, prepositional phrases, passive constructions, expletive sentences, verb-particle constructions, auxiliaries, light verb constructions, etc. Most of the new rules are discussed in more details in (Abzianidze, 2015).

[5] SICK is partitioned in three parts (trail, train and test) and used as a benchmark for RTE14 (Marelli et al., 2014a). The data and the system results of RTE14 are available at http://alt.qcri.org/semeval2014/task1/

[6] Automating a tableau rule extraction is quite hard for the following reasons: it is unclear how to determine automatically whether a CCG derivation is wrong, a tableau rule is missing, or lexical knowledge is lacking; and the general format of a rule makes search procedure extremely inefficient.

ID	Gold/LP	Problem (premise ? conclusion)
3670	E/N	It is raining on a walking man ? A man is walking in the rain
219	E/N	There is no girl in white dancing ? A girl in white is dancing
5248	N/E	Someone is playing with a toad ? Someone is playing with a frog
8490	N/C	A man with a shirt is holding a football ? A man with no shirt is holding a football
7402	N/C	There is no man and child kayaking through gentle waters ? A man and a young boy are riding in a yellow kayak
1431	C/C	A man is playing a guitar ? A man is not playing a guitar
8913	N/C	A couple is not looking at a map ? A couple is looking at a map

Table 1: Problems from SICK-trial and SICK-train with gold and LangPro judgments.

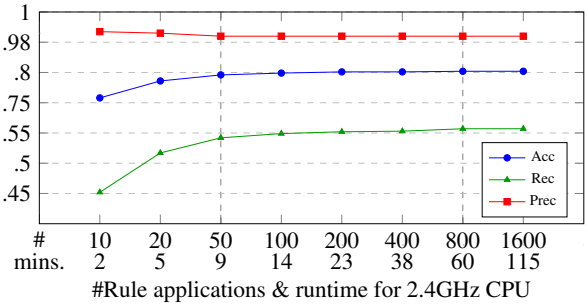

Figure 6: Performance of LangPro on SICK-train (4500) using CCG derivations of the C&C parser.

LangPro \ SICK	test (4927 problems)		
	Prec%	Rec%	Acc%
Baseline (majority)	-	-	56.36
+C&C+50	98.03	53.75	79.52
+EasyCCG+50	98.03	51.41	78.53
LangPro Hybrid-50	97.99	57.03	80.90
+C&C+800	97.99	54.73	79.93
+EasyCCG+800	98.00	52.67	79.05
LangPro Hybrid-800	97.95	58.11	**81.35**

Table 2: Evaluation of the versions of LangPro

LangPro was unable to prove several problems requiring complex background knowledge (e.g., SICK-3670 in Table 1) or having wrong CCG derivations from the C&C parser (e.g., in SICK-219, *white dancing* is a noun constituent).

5.2 Development

The part of the SICK data, SICK-train, issued for training at RTE14 was used for development. After running LangPro on SICK-train, we only analyzed false positives, i.e. neutral problems that were identified either as entailment or contradiction by the prover. The analysis reveals that the parsers and WordNet are responsible for almost all these errors. For example, in Table 1, SICK-5248 is classified as entailment since *toad* and *frog* might have synonymous senses; this problem shows the advantage of not using WSD, where a proof search also searches for word senses that might give rise to a logical relation. SICK-7402 was falsely identified as contradiction because of the wrong analyses of the premise by both CCG parsers: *no man* and *child...* are in coordination, which implies *there is no man*, and hence, contradicts the conclusion. SICK-8490 is proved as contradiction since the prover considers LLFs where *shirt* takes a wide scope. With the help of LangPro, we also identified inconsistency in the annotations of problems, e.g., SICK-1431, 8913 are similar problems but classified differently; it is also surprising that SICK-5248 is classified as neutral.

During this phase, also the effective (800) and efficient (50) upper bounds for the rule application number were determined (see Figure 6). Moreover, 97.4% of proofs found in 1600 rule applications are actually attainable in at most 50 rule applications; this shows that the rule application strategy of LangPro is quite efficient.

5.3 Evaluation

We evaluate LangPro on the unseen portion of the SICK data, SICK-test, which was used as a benchmark at RTE14; the data was also held out from the process of designing LLFgen. The prover classifies each SICK problem as follows:

input: (T, H);
try t = the first LLF of `llf`(T);
h = the first LLF of `llf`(H)
if no error **then**
case `tab`$\{t{:}\mathbb{T}, h{:}\mathbb{F}\}$, `tab`$\{t{:}\mathbb{T}, h{:}\mathbb{T}\}$
CLOSED, OPEN: classify as `ENTAILMENT`;
OPEN, CLOSED: classify as `CONTRADICTION`;
OPEN, OPEN: classify as `NEUTRAL`;
CLOSED, CLOSED: classify as `ENTAILMENT`; report it;
else classify as `NEUTRAL`; report it;

The results, in Table 2, show evaluation of LangPro on SICK-test using both parsers separately with the efficient and effective rule application upper bounds. Slightly better results with the C&C parser is explained by employing the parser in the learning phase. The difference of .5% in

System \ Measure+	Prec%	Rec%	Acc% (+LP)
Illinois-LH	81.56	**81.87**	**84.57** (+0.55)
ECNU	84.37	74.37	83.64 (+1.69)
UNAL-NLP	81.99	76.80	83.05 (+1.44)
SemantiKLUE	85.40	69.63	82.32 (+**2.78**)
The Meaning Factory	93.63	60.64	81.59 (+2.72)
LangPro Hybrid-800	**97.95**	58.11	81.35
UTexas	97.87	38.71	73.23 (+8.97)
Prob-FOL	-	-	76.52
Nutcracker	-	-	78.40
Baseline (majority)	-	-	56.69

Table 3: Comparing LangPro to the top or related RTE systems and combining their answers[7]

accuracy between the C&C-based and EasyCCG-based provers show that LLFgen was not fitted to the C&C parser's output during the learning phase.

In order to eliminate at some extent errors coming from the parsers, hybrid provers are designed that simply combine answers of two systems—if one of the systems proves a relation then it is an answer. Both hybrid versions of LangPro show more than 80% of accuracy while only 5 systems were able to do so at RTE14, where 77.1% was a median accuracy. The prover turns out to be extremely reliable with its state-of-the-art precision being almost 98%. A high precision is conditioned by the formal deductive proof nature of LangPro and by the sound rules it employs.

In Table 3, we compare the best version of hybrid LangPro to the top 5 systems of RTE14 on SICK-test and show the improvement it gives to each system when blindly adopting its positive answers (i.e. entailment and contradiction).

The decision procedure of the prover is completely rule-based and easy to comprehend since it follows the intuitive deductive rules. Tableaux proofs by LangPro for SICK-247 (in Figure 7) and SICK-2895 (in Figure 8) show step by step how T contradicts and entails, respectively, H.[8] Several new rules employed in these tableaux are given in Figure 9. Note that the both problems, SICK-247, 2895, were wrongly classified by all the top 7 systems of the RTE14. Taking into account that solving SICK-247 requires a sort of De Morgan's law for negation and disjunction, this demonstrates where LangPro, a purely logic-based system, outperforms non-logic-based systems.[9] The another problem, SICK-2895, is an evidence how unreliable the state-of-the-art and non-logic-based RTE systems might be since solving the problem only requires a lexical knowledge $barbell \leq weight$, which is available in WordNet.

6 Related Work

Using formal logic tools for a wide-coverage RTE task goes back to the Nutcracker system (Bos and Markert, 2005), where a wide-coverage semantic processing tool Boxer (Bos, 2008), in combination with the C&C tools, first produces discourse representation structures of (Kamp and Reyle, 1993) and then FOL semantic representations (Curran et al., 2007). Reasoning over FOL formulas is carried by off-the-shelf theorem provers and model builders for FOL.[10] Our approach differs from the latter in several main aspects: (i) the underling logic of LLFs (i.e. higher-order logic) is more expressive than FOL (e.g., it can properly model GQs and subsective adjectives), (ii) LLFs are cheap to get as they are easily obtained from CCG derivations, and (iii) we develop a completely new proof procedure and a prover for a version of Natural Logic.

The other related works are (MacCartney and Manning, 2008) and (Angeli and Manning, 2014). Both works contribute to Natural Logic and are based on the same methodology.[11] The approach has two main shortcomings compared to Natural Tableau; namely, it is unable to process multi-premised problems, and its underling logic is weaker (e.g., according to (MacCartney, 2009), it cannot capture the entailment in Figure 2).

[7]The top 5 systems of RTE14 are Illinois-LH (Lai and Hockenmaier, 2014), ECNU (Zhao et al., 2014), UNAL-NLP (Jimenez et al., 2014), SemantiKLUE (Proisl et al., 2014) and The Meaning Factory (Bjerva et al., 2014).

[8]In the tableaux, due to lack of space, several constants are denoted with initial characters of their lemmas and some intermediate nodes are omitted. Some of the nodes are annotated with a sequence of source rule applications.

[9]Even a shallow heuristic—if H has a named entity that does not appear in T, then there is no entailment—is not sufficient for showing that SICK-247 is contradiction. We thank our reviewer for mentioning this heuristic w.r.t. SICK-247.

[10]Nutcracker obtains 3% lower accuracy on SICK than our prover (Pavlick et al., 2015). The Meaning Factory (Bjerva et al., 2014) that is a brother system of Nutcracker, instead of solely relying on decisions of theorem provers and model builders, uses machine learning methods over the features extracted from these tools; this method results in a more robust system. RTE systems UTexas (Beltagy et al., 2014) and Prob-FOL (Beltagy and Erk, 2015) also use Boxer FOL representations but employ probabilistic FOL. For comparison purposes, the results of these systems on the SICK data are given in Table 3.

[11]They relate two sentences by a sequence of string edits; the final logical relation between the sentences is computed by composing logical relations associated with these edits.

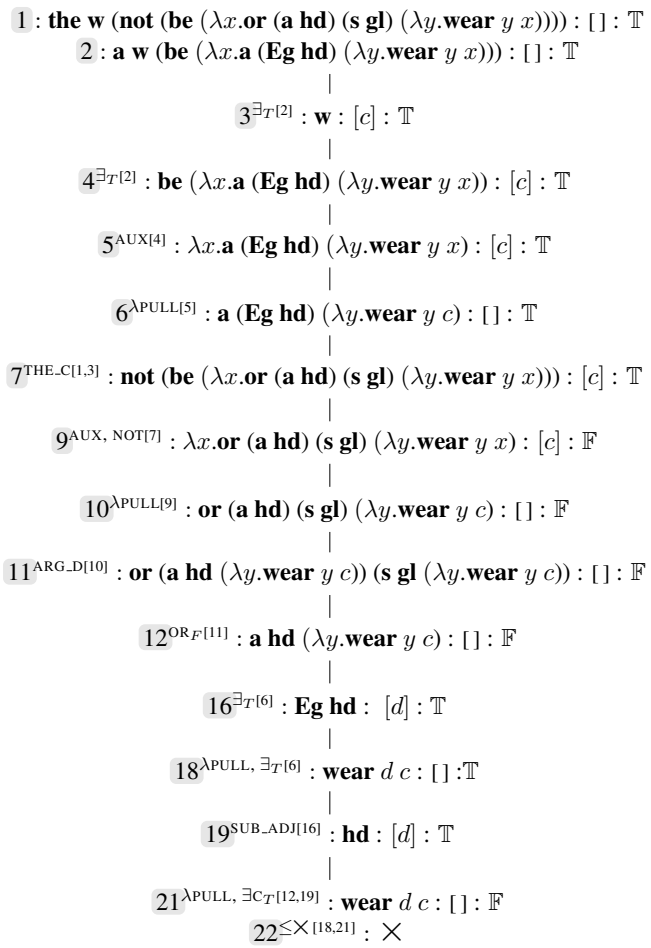

Figure 7: A closed tableau for SICK-247: *The woman is not wearing glasses or a headdress* $\perp$ *A woman is wearing an Egyptian headdress*

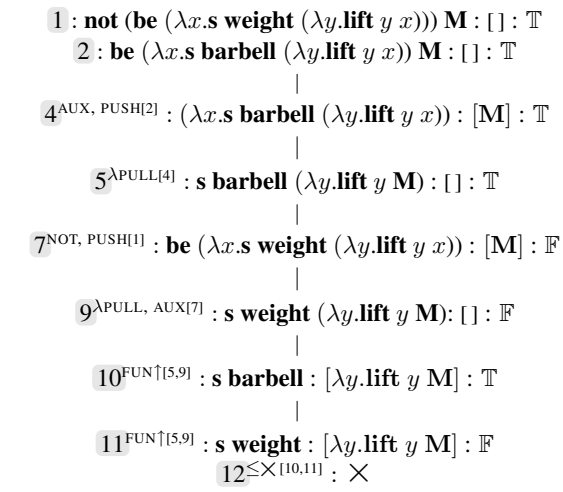

Figure 8: A closed tableau for SICK-2895: *The man isn't lifting weights* $\perp$ *The man is lifting barbells*, where M abbreviates a shared term **the man** aligned by the LLF-aligner.

$$\frac{G\ A : [\vec{C}] : \mathbb{T} \quad H\ A : [\vec{C}] : \mathbb{F}}{\begin{array}{l} G : [A, \vec{C}] : \mathbb{T} \\ H : [A, \vec{C}] : \mathbb{F} \end{array}}\text{ARG} \qquad \frac{F\ A : [\vec{C}] : \mathbb{T} \quad F\ B : [\vec{C}] : \mathbb{F}}{\begin{array}{l} A : [\vec{D}] : \mathbb{T} \\ B : [\vec{D}] : \mathbb{F} \\ \text{s.t. } \vec{D} \text{ is fresh and} \\ F \text{ is upward monotone} \end{array}}\text{FUN}\uparrow$$

$$\frac{\mathbf{the}\ N\ V : [\,] : \mathbb{X} \quad N : [c] : \mathbb{T}}{V : [c] : \mathbb{X}}\text{THE\_C} \qquad \frac{A\ N : [c] : \mathbb{T}}{\begin{array}{l} N : [c] : \mathbb{T} \\ \text{s.t. } A \text{ is subsective} \end{array}}\text{SUB\_ADJ}$$

$$\frac{X\ V : [\vec{C}] : \mathbb{X}}{\begin{array}{l} V : [\vec{C}] : \mathbb{X} \\ \text{s.t. } X \in \{\mathbf{be}, \mathbf{do}\} \end{array}}\text{AUX} \qquad \frac{\lambda x.\ A : [d\vec{C}] : \mathbb{X}}{A[x := d] : [\vec{C}] : \mathbb{X}}\lambda\text{PULL}$$

Figure 9: Several rules learned from SICK-trial

7 Conclusion and Future Work

We made Natural Tableau of Muskens (2010) suitable for the wide-coverage RTE task by extending it both in terms of rules and language. Based on the extended Natural Tableau, the prover LangPro was implemented, which has a modular architecture consisting of the inventory of rules, KB and the LLF generator. As a whole, the prover represents a deductive model of natural reasoning with the transparent and naturally interpretable decision procedure. While learning only from the SICK-trial data, LangPro showed the comparable accuracy and the state-of-the-art precision on the unseen SICK data.

For future work, we plan to explore the FraCaS (Consortium et al., 1996) and newswire RTE (Dagan et al., 2005) data sets to further improve the LLF generator and enrich the inventory of tableau rules. These tasks are also interesting for two reasons: to find out how much effort is required for adapting the LLF generator to different data, and which rules are to be added to the inventory for tackling the new RTE problems. Incorporating more WordNet relations (e.g., similarity, derivation and verb-group) and the paraphrase database (Ganitkevitch et al., 2013) in KB is also a part of our future plans.

Acknowledgments

I thank Reinhard Muskens for helpful discussions on this work, Jan Sprenger for his feedback on an early version of this paper, the authors of (Honnibal et al., 2010) for sharing a recent parser model, and anonymous reviewers for useful comments. This work is a part of the project "Towards Logics that Model Natural Reasoning" and supported by the NWO grant (project number 360-80-050).

References

Lasha Abzianidze. 2015. Towards a wide-coverage tableau method for natural logic. In Tsuyoshi Murata, Koji Mineshima, and Daisuke Bekki, editors, *New Frontiers in Artificial Intelligence*, volume 9067 of *Lecture Notes in Artificial Intelligence*, page Forthcoming. Springer-Verlag Berlin Heidelberg.

Gabor Angeli and Christopher D. Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–545, Doha, Qatar, October. Association for Computational Linguistics.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4(2):159–219.

Islam Beltagy and Katrin Erk. 2015. On the proper treatment of quantifiers in probabilistic logic semantics. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS-2015)*, London, UK, April.

Islam Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond Mooney. 2014. Utexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 796–801, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Evert W. Beth. 1955. Semantic Entailment and Formal Derivability. *Koninklijke Nederlandse Akademie van Wentenschappen, Proceedings of the Section of Sciences*, 18:309–342.

Johannes Bjerva, Johan Bos, Rob Van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646, Dublin, Ireland.

Patrick Blackburn and Johan Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Press.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 628–635.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

Johan Bos. 2009. Towards a large-scale formal semantic lexicon for text processing. In C. Chiarcos, R. Eckart de Castilho, and Manfred Stede, editors, *From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennal GSCL Conference 2009*, pages 3–14.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33.

The Fracas Consortium, Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. 1996. Using the framework.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic, June. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Marcello D'Agostino, Dov M. Gabbay, Reiner Hhnle, and Joachim Posegga, editors. 1999. *Handbook of Tableau Methods*. Springer.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Melvin Fitting. 1990. *First-order Logic and Automated Theorem Proving*. Springer-Verlag New York, Inc., New York, NY, USA.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Comput. Linguist.*, 33(3):355–396, September.

Matthew Honnibal, James R. Curran, and Johan Bos. 2010. Rebanking ccgbank for improved np interpretation. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 207–215, Uppsala, Sweden.

Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of*

the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 732–742, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Hans Kamp and Uwe Reyle. 1993. *From discourse to logic; an introduction to modeltheoretic semantics of natural language, formal logic and DRT*. Dordrecht: Kluwer.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

George Lakoff. 1972. Linguistics and natural logic. In Donald Davidson and Gilbert Harman, editors, *Semantics of Natural Language*, volume 40 of *Synthese Library*, pages 545–665. Springer Netherlands.

Mike Lewis and Mark Steedman. 2014. A* ccg parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar, October. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In Donia Scott and Hans Uszkoreit, editors, *COLING*, pages 521–528.

Bill MacCartney. 2009. *Natural language inference*. Phd thesis, Stanford University.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of SemEval 2014 (International Workshop on Semantic Evaluation)*, pages 1–8, East Stroudsburg PA. ACL.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223, September.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Richard Montague. 1974. English as a Formal Language. In Richmond H. Thomason, editor, *Formal philosophy: Selected Papers of Richard Montague*, chapter 6, pages 188–221. Yale University Press.

Reinhard Muskens. 2010. An analytic tableau system for natural logic. In Maria Aloni, Harald Bastiaanse, Tikitu de Jager, and Katrin Schulz, editors, *Logic, Language and Meaning*, volume 6042 of *Lecture Notes in Computer Science*, pages 104–113. Springer Berlin Heidelberg.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. Adding semantics to data-driven paraphrasing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 1512–1522.

Thomas Proisl, Stefan Evert, Paul Greiner, and Besim Kabashi. 2014. Semantiklue: Robust semantic similarity at multiple levels using maximum weight matching. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 532–540, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In D. Borsley, Robert and Kersti Brjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, pages 181–224. Wiley-Blackwell.

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

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.