SOLVING TEXTUAL ENTAILMENT WITH THE THEOREM PROVER FOR NATURAL LANGUAGE

Lasha Abzianidze

CLCG, University of Groningen, the Netherlands
L.Abzianidze@rug.nl

Abstract

We present a theorem prover for natural language and show how it processes various types of textual entailment problems. The prover itself is based on a tableau system for natural logic that employs logical forms similar to linguistic expressions. With respect to the problems drawn from textual entailment datasets, a wide-range of the judgments of the prover are discussed, including both correct and incorrect ones. The analysis shows that the false proofs, which are extremely rare, are mainly due to the wrong lexical senses or the noisy gold labels of the dataset. Knowledge sparsity is identified as the main reason for the failure in proof search.

Keywords and phrases: Analytic tableau method, theorem proving, natural reasoning, lambda logical form, natural logic, typed lambda calculus, textual entailment, combinatory categorial grammar

1 Introduction

The automatic detection of logical relations between natural language expressions is a fundamental problem of natural language understanding. In order to model the human reasoning over natural language text, textual entailment data is used for training and evaluating the theories and systems. The data represents a collection of text pairs annotated on the entailment, contradiction and neutral relations by humans. The annotated textual entailments are further used as a benchmark in the recognizing textual entailment (RTE) challenges [8]. On the other hand, studies on formal semantics seek formal logics that model linguistic semantics. Such formal logics supported by an automated theorem prover usually leads to an automated reasoner for natural language. A notable example of this research line is the RTE system NutCracker [6] which combines first-order logic and theorem proving to account for the natural reasoning. We follow the latter research line by adopting a version of higher-order logic as a semantic representation and employing an analytic tableau system for theorem proving. The combination results in an implemented theorem prover for natural language that is applicable to wide-coverage text using a robust syntactic parser as a preprocessor.

While our previous works [2, 1, 3] talks about the theorem prover for natural language, it was never the purpose to show the matters of failure and success of the prover with respect to entailment problems. Taking into account that the prover has almost prefect precision ($\approx 98\%$) with competitive accuracy [1], it is interesting to see the false proofs or to find out what is the reason for relatively small recall (61%). In this paper we explore these cases by considering a plethora of entailment problems that get mixed judgments from the prover. Based on the analysis, we give the rationale for the decisions of the prover and draw the directions towards improvement of the performance.

The rest of the paper is structured as follows. First, the theory behind the prover, the natural tableau system [22], and its extensions are introduced. Then two relevant components of the natural logic theorem prover are described. The natural logic theorem prover is followed by the prover for natural language. We outline its architecture and the details of entailment processing. For each classification type we discuss several entailment problems that illustrate the issues of theorem proving. Based on the examples, the reasons for true, false and failed proofs are explained. The paper ends with final remarks and a description of future work.

2 Natural tableau system

An analytic tableau system for natural logic [22] is a proof procedure over the logical forms of natural language expressions. The logical forms, so-called Lambda Logical Forms (LLFs), are merely typed λ-terms that employ lexical constants and no logical connectives.[1] LLFs resemble the surface forms and can be considered as formulas of some sort of *natural logic* [15, 5, 26], hence we refer the tableau system as the *natural tableau*. On the other hand, a tableau method is a refutation proof procedure [10]. To prove a statement, it attempts to find a situation that serves as a counterexample to the statement. If such a situation is found, the statement is disproved; otherwise it is proved. The search for a situation is done by building a tableau, also called a truth tree. Each branch of a tableau represents a situation. The construction of a tableau is guided by a predefined set of tableau rules.

[1]In contrast to [22], throughout the paper we assume the extensional semantic types built upon the entity e and truth t types while the type s for possible worlds is dropped out. The terms for common lexical elements are typed in an ordinary way: nouns and intransitive verbs as et, transitive verbs as eet, proper names as e, quantifiers as $(et)(et)t$.

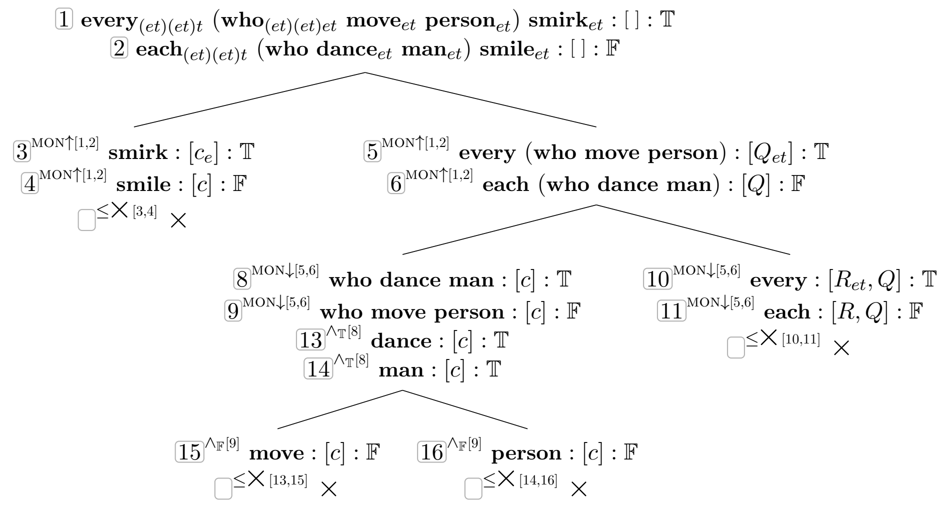
1 every_(et)(et)t (who_(et)(et)et move_et person_et) smirk_et : [ ] : 𝕋
2 each_(et)(et)t (who dance_et man_et) smile_et : [ ] : 𝔽
3 MON↑[1,2] smirk : [c_e] : 𝕋
4 MON↑[1,2] smile : [c] : 𝔽
≤× [3,4] ×
5 MON↑[1,2] every (who move person) : [Q_et] : 𝕋
6 MON↑[1,2] each (who dance man) : [Q] : 𝔽
8 MON↓[5,6] who dance man : [c] : 𝕋
9 MON↓[5,6] who move person : [c] : 𝔽
13 ∧𝕋[8] dance : [c] : 𝕋
14 ∧𝕋[8] man : [c] : 𝕋
10 MON↓[5,6] every : [R_et, Q] : 𝕋
11 MON↓[5,6] each : [R, Q] : 𝔽
≤× [10,11] ×
15 ∧𝔽[9] move : [c] : 𝔽
≤× [13,15] ×
16 ∧𝔽[9] person : [c] : 𝔽
≤× [14,16] ×

Figure 1. The closed tableau represents a failed attempt to refute that the TEP P:"*every person who moves smirks*", C:"*each man who dances smiles*" does not represent entailment. For the first appearance of a term, its type is written in a subscript and assumed for its later occurrences. Each node is labeled with an ID and a source rule application (i.e. the rule and the IDs of the nodes it applies).

In order to illustrate how the natural tableau works, we give a tableau in Fig. 1 which refutes a given textual entailment problem (TEP) that does not represent entailment. The tableau starts with the counterexample, $\boxed{1}$ and $\boxed{2}$ nodes, of the argument.[2] It is expanded using the MON↑ rule (see Fig. 2) which takes into account upward monotonicity (mon↑) of function terms. MON↑ is a branching rule and its application yields two situations. The left branch corresponds to the situations where $A \not\leq B$ holds,[3] and the right one to the situations where $G \not\leq H$ holds, including those where $A \leq B$ holds. Since **every** and **each** are mon↑ in the second argument position, (MON↑) applies to them—its antecedents match the nodes and its consequents are introduced in the tableau. From the resulting branches, the left one is closed

[2]A tableau entry (i.e. node) represents a pair of an LLF and its argument list which is additionally marked with a truth sign. The semantics behind an entry is that the term obtained by applying the LLF to its arguments is evaluated according to the sign. The truth signs $\mathbb{T}$ (true) and $\mathbb{F}$ (false) represent the terms of type t. Annotations on the nodes give information about the building process of a tableau. $\vec{C}$ denotes a sequence of the terms.

[3]For any A_α and B_α lexical terms of the same type α, we say B *contains* A and write as $A \leq B$ iff $\forall \vec{X}(A\vec{X} \rightarrow B\vec{X})$, where $\mathbb{F} \rightarrow \mathbb{T}$. In this way, $\mathbf{dog}_{et} \leq \mathbf{animal}_{et}$ holds since based on the lexical knowledge $\forall x_e(\mathbf{dog}\, x_e \rightarrow \mathbf{animal}\, x_e)$ holds. Notice that in case of truth values the containment relation $\leq$ coincides with the material implication $\rightarrow$.

$$\dfrac{\begin{array}{c} GA : [\vec{C}] : \mathbb{T} \\ HB : [\vec{C}] : \mathbb{F} \end{array}}{\begin{array}{cc} A : [\vec{d}\,] : \mathbb{T} & G : [Q, \vec{C}] : \mathbb{T} \\ B : [\vec{d}\,] : \mathbb{F} & H : [Q, \vec{C}] : \mathbb{F} \end{array}}\,\textsc{mon}\uparrow \qquad \dfrac{\begin{array}{c} GA : [\vec{C}] : \mathbb{T} \\ HB : [\vec{C}] : \mathbb{F} \end{array}}{\begin{array}{cc} A : [\vec{d}\,] : \mathbb{F} & G : [Q, \vec{C}] : \mathbb{T} \\ B : [\vec{d}\,] : \mathbb{T} & H : [Q, \vec{C}] : \mathbb{F} \end{array}}\,\textsc{mon}\downarrow \qquad \dfrac{\begin{array}{c} A : [\vec{C}] : \mathbb{T} \\ B : [\vec{C}] : \mathbb{F} \end{array}}{\times}\leq\times$$

$$\begin{array}{c}\text{where } G \text{ or } H \text{ is mon}\uparrow \\ \text{and } \vec{d} \text{ and } Q \text{ are fresh}\end{array} \qquad \begin{array}{c}\text{where } G \text{ or } H \text{ is mon}\downarrow \\ \text{and } \vec{d} \text{ and } Q \text{ are fresh}\end{array} \qquad \text{where } A \leq B$$

Figure 2. The tableau rules for monotonic operators

as it is identified as inconsistent with the help of the closure ($\leq \times$) rule in Fig. 2. The right branch is further developed by applying ($\textsc{mon}\downarrow$) to $\boxed{5}$ and $\boxed{6}$. The rule application takes into account the downward monotonicity (mon↓) of **every** and **each** in the first argument position. From the new branches, the right one is closed due to inconsistency while the left one is grown from $\boxed{8}$ and $\boxed{9}$ with the rules that treats **who** as a conjunction between terms of type *et*. In the end, each branch of the tableau is closed, i.e. the tableau is closed, which indicates the failure in refuting the textual entailment; therefore the proof for the entailment relation is found.

The previous works [2, 1] extend the natural tableau in several directions in order to make it viable beyond toy examples. First, we incorporate the following syntactic types in the type system: `n` for nouns, `np` for noun phrases, `s` for sentences and `pp` for prepositional phrases. From the perspective of theorem proving, LLFs with syntactic types offer fine-grained matching between tableau entries and antecedents of the rules. For instance, an LLF of type *et* can ambiguously correspond to a noun or an intransitive verb. This ambiguity needs to be resolved before applying a rule; this complicates a proof procedure. On the other hand, an LLF of syntactic type contains no such ambiguity. Interaction between the syntactic and semantics types is established by the subtyping $\sqsubseteq$ relation defined as:

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

(b) for any $\alpha_1, \alpha_2, \beta_1, \beta_2$ types, $(\alpha_1, \alpha_2) \sqsubseteq (\beta_1, \beta_2)$ iff $\beta_1 \sqsubseteq \alpha_1$ and $\alpha_2 \sqsubseteq \beta_2$

An LLF with syntactic types still has the similar form as its semantic counterpart. For instance, compare the LLF in (1), where `vp` abbreviates (`np`, `s`), to the one in $\boxed{1}$ of Fig. 1.

$$\mathbf{every}_{\mathtt{n},\mathtt{vp},\mathtt{s}}\ (\mathbf{who}_{\mathtt{vp},\mathtt{n},\mathtt{n}}\ \mathbf{move}_{\mathtt{vp}}\ \mathbf{person}_{\mathtt{n}})\ \mathbf{smirk}_{\mathtt{vp}} \tag{1}$$

$$\mathtt{modifierSet} : \mathtt{LLF} : \mathtt{argList} : \mathtt{truthSign} \tag{2}$$

Moreover, for a single lexical element it is rarely necessary to have two lexical terms with semantic and syntactic types. This is facilitated by the subtyping relation as $\textbf{smile}_{\texttt{vp}}c_e$ and $\textbf{person}_{\texttt{n}}c_e$ terms are well-formed and there is no need for the semantic terms $\texttt{smile}_{et}$ and $\texttt{person}_{et}$.

The second extension to the system is the introduction of a modifier set in tableau entries, see (2). As argued in [2], the set is used for saving and later retrieving a modifier term that is indirectly applied to its head. This technique is used for the nouns with several adnominals or for the verbs with several adverbs. The trick with the modifier set solves the problem of event modification without introducing an event variable in an LLF and losing the *essence* of natural logic.[4]

The last extension is in terms of tableau rules. While [22] presents the rules for monotonic operators, Boolean connectives, quantifiers, etc., the natural tableau lacks the rules for many phenomena that are often found in open domain text. The inventory of rules is further enriched with the rules for nominal and verbal modifiers, prepositions, copula, passive constrictions, expletives, etc. The procedure of collecting the rules is described in [1]; the rules are presented in [2].

3 Natural logic theorem prover

In order to automatize the natural tableau system, we implemented the theorem prover for natural logic, called NatPro. The prover is written in the Prolog language. It expects a finite set of signed LLFs as an input and, using the inventory of rules and the knowledge base, tries to build a tableau over the input. Below we describe the components of the prover. The organization of the knowledge base is explored in more details as it is relevant for certain judgments discussed in Sec. 5.

3.1 Knowledge base (KB)

Importance of background knowledge for open domain textual entailment is extremely high. Background knowledge is of two kinds. One is *extra-linguistic knowledge*, also called *encyclopedic* or *world knowledge*, which mainly involves information how the current state of the situation or world is organized. For example, correct classification of the TEPs (K1) and (K2) in Table 1 require extra-linguistic knowledge. Another type of knowledge is *linguistic* which encompasses the lexical knowledge and its compositions, where composition is governed by the grammar. In other words, this type of knowledge is encoded in the language itself. For instance, the linguistic

[4]Notice that the extension of the natural tableau is conservative—the tableaux generated with the system of [22], e.g., the one in Figure 2, are still available in the extended version.

Table 1. Examples of textual entailment problems

ID	Premise	Conclusion
K1	Barcelona defeated Real Madrid 4-0	Barcelona thumped Real Madrid
K2	Messi won Ballon d'Or	Lionel Messi won the Ballon d'Or award
K3	Not all birds fly	There is some bird that does not fly
K4	The piano is being played by the boy	The boy is playing a musical instrument
S5	A sad girl is crying	A girl is weeping
S6	The child is crying	The child is screaming and weeping
S7	The child is screaming	The child is weeping

knowledge is required for solving the TEPs (K3) and (K4) in Table 1. Despite this distinction the border between the linguistic and extra-linguistic knowledge is quite vague.

Currently we model only the linguistic knowledge in the tableau system by taking WordNet [12] as a lexical knowledge base. At this moment, for simplicity we employ only the hypernymy and antonymy relations of WordNet. Using these relations, we define the containment ($\leq$) and disjoint ($|$) relations over lexical terms A and B as follows. $A \leq B$ holds if there exist the WordNet synsets S_A and S_B such that S_B is at least as general as S_A and some senses of A and B are in S_A and S_B, respectively. According to this definition, $\mathbf{cry}_{\mathtt{vp}} \leq \mathbf{scream}_{\mathtt{vp}}$ and $\mathbf{scream}_{\mathtt{vp}} \leq \mathbf{cry}_{\mathtt{vp}}$ because there exists senses of $\mathbf{cry}_{\mathtt{vp}}$ and $\mathbf{scream}_{\mathtt{vp}}$ that belong to the same synset with the meaning of "*utter a sudden loud cry*". Similarly, A and B terms are disjoint, written as $A\,|\,B$, if some of their senses belong to the synsets that are in the antonymy relation. In this way, $\mathbf{empty}_{\mathtt{n,n}} \mid \mathbf{full}_{\mathtt{n,n}}$ holds as there exist antonymous singleton synsets with the senses related to the terms. This kind of treatment of lexical semantics assumes that each lexical term has multiple senses independently from the context it occurs in. Due to this feature we call it the *multi-sense* approach.

The proof search with multiple senses can be interpreted as a search over lexical senses too: are there senses for the lexical terms that licenses the given TEP as entailment or contradiction? For example, proving equivalence of the sentences in (S5) literally means that based on certain lexical semantics the sentences are equivalent. The multi-sense approach has a shortcoming as it validates the entailment relations like one in (S6). But it does not lead to any relation between "*weep*" and "*scream*" since their senses are not related in WordNet, hence there is no logical relation between the sentences in (S7).

Compared to the sense disambiguated approach, with multiple senses it is more likely that there is a relation between two terms. Taking into account that rule-based RTE systems often have problems related to knowledge sparsity, the multi-sense approach seems an interesting option for the tableau prover to overcome the sparsity to some extent. Moreover, the multi-sense approach is simpler and more robust since it does not require to disambiguate word senses, the latter representing an open problem in NLP.[5]

3.2 Inventory of the rules (IR)

The inventory of the rules is the most crucial component of the tableau system. The tableau rules encode simple reasoning steps and instructions how to decompose large LLFs into smaller pieces. According to the phenomena the rules account for, they can be roughly categorized into two groups. The first group of the rules are mainly the ones presented in [22]. These rules unfold the semantics of the LLFs involving the terms with the algebraic properties: Boolean, upward monotone, downward monotone, etc. Hence the rules of this group are called *algebraic*. The rules for determiners and those concerning the format of tableau entries are also part of the algebraic rules. This group also contains so-called *admissible* rules—the rules that are redundant from a completeness point of view but represent a shortcut for several rule applications. Many algebraic rules either introduce a fresh entity, employ an old entity or has a branching structure; these make the rules inefficient from the theorem proving perspective. The admissible rules can be seen as a way of applying some of the inefficient rules in an efficient manner.

The second group counts the rules that essentially deal with LLFs modeling common syntactic constructions; let them be *syntactic* rules. The syntactic rules analyze adjective-head and adverb-head pairs, prepositional phrases, passive constrictions, compound nouns, the constructions with a copula, auxiliary and light verbs, etc.; most of these rules are described in [2].

3.3 A proof engine (PE)

In the prover, a tableau is represented as a list of tableau branches, where a branch itself is a list of signed LLFs. For terminating a tableau building process in a finite time, we set the limit for the number of rule applications (RAL). If there is an open branch after the RAL is reached, the tableau is considered open. In order to make sure that each branch gets its fair share of rule applications, after each rule application on a branch, its next branch

[5]The upper limit of a word sense disambiguation (WSD) system with respect to the WordNet senses (i.e. fine-grained senses) is quite low as the inter-annotator agreement is only 72.5% [24]. Even for course-grained senses the inter-annotator agreement is only 86.5% [23].

is processed; in case of the last branch, the first branch is processed. In average, this strategy guarantees finding an open branch, if it exists, earlier than the strategy where a shift to a new branch happens only if the current one closes.

The *proof engine* of the prover represents an algorithm that decides which of the rule applications is to be carried out next. For the rule application strategy it takes into account *efficiency* of each tableau rule, where efficiency depends on the properties of a rule, e.g, whether it is branching, employs old constants or produces fresh ones. For example, (MON$\uparrow$) and (MON$\downarrow$) from Fig. 2 are inefficient rules as they are branching and introduce fresh constants. These rules also have an interesting feature: their antecedents are not equivalent to the disjunction of the consequents. So, discarding a node from a branch after applying such a *non-equivalent* rule to the node might lead to incompleteness. In light of the non-equivalent rules, the proof engine needs to track down a rule application history for each branch in order to avoid performing the same rule application more than once. Recording the history is also important for the admissible rules.

4 LangPro: natural language theorem prover

The section describes the architecture of the theorem prover that reasons over natural language expressions. It is also illustrated how the prover operates on a certain TEP. In the end, the evaluation results of the prover on RTE datasets are presented.

4.1 The architecture

A theorem prover for natural language, called LangPro, is obtained by combining a CCG parser[6], the LLF generator [1] and the natural logic theorem prover (see Fig. 3). In case of the parser component we have at least two choices: the C&C [7] and EasyCCG [16] parsers.[7] It is interesting to employ both parsers as they are based on different approaches. Hereafter, the version of LangPro based on C&C or EasyCCG is referred as ccLangPro or easyLangPro, respectively.

[6]In LLFs, lexical elements are interpreted as functions. Since this interpretation is fundamental for categorial grammars (CGs), the CG-style derivation is a good starting point for obtaining LLFs. To the best of our knowledge, the only wide-coverage CG-based parsers analyze sentences in Combinatory Categorial Grammar (CCG) [25].

[7]In the current settings of LangPro, we employ C&C with the model trained on the improved corpus [13]. While EasyCCG acts as a multi-parser, returning n-best derivations, currently only the first best derivation is employed.

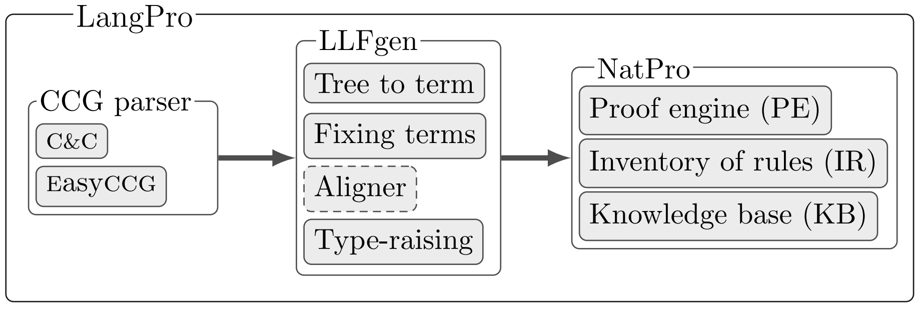

Figure 3. The architecture of LangPro

While describing the architecture of LangPro, in Fig. 4 we also demosntrate how the prover operates on a particlar TEP—SICK-1417 from the SICK dataset [21]. First, the premise and the conclusion are parsed by a parser. The obtained CCG derivation trees are processed by the LLF generator (LLFgen): first, the derivations are converted into CCG terms by removing directionality from the CCG categories, then the terms are *corrected* and further transformed into LLFs by type-raising the quantified NPs.[8] Depending on the choice of the parser, we call an obtained LLF a ccLLF or easyLLF.

Often a premise and a conclusion share several multiword phrases, analysis of which is not relevant for the classification. Due to this reason, many RTE systems adopt alignment techniques that help the systems to concentrate on relevant parts of the text [9]. In LangPro alignment is carried out by the term aligner, a part of LLFgen. The aligner finds the subterms occurring in both CCG terms and replaces them with fresh terms. While doing so, the candidate subterms are checked for monotonicity: they are aligned iff they are not mon$\downarrow$ (for more details see SICK-1207 in Sec. 5). Sometimes the alignment procedure leaves the CCG terms unchanged, like in the current example. Since the alignment procedure may eliminate the chance of finding a proof, the original LLFs are tested if the prover is not able to find a proof with the aligned LLFs. Henceforth, $\overline{\text{cc}}$LLFs and $\overline{\text{easy}}$LLFs will denote the aligned versions of the corresponding LLFs. The aligner is an optional component of the prover which contributes to short proofs, hence to the prefromance too [3].

For each CCG derivation, LLFgen returns a list of LLFs. In the running

[8]The CCG terms are typed with the syntactic types corresponding to the CCG categories. They are not well-formed λ-terms due to the remains of the *type changing* (i.e. *lexical*) combinatory rule of the CCG parsers. The CCG terms are fixed with term-rewriting rules which mainly explain the type changes (see Fig. 4) or modify the terms for better semantic adequacy. More details about the LLFgen procedures can be found in [1, Sec. 3].

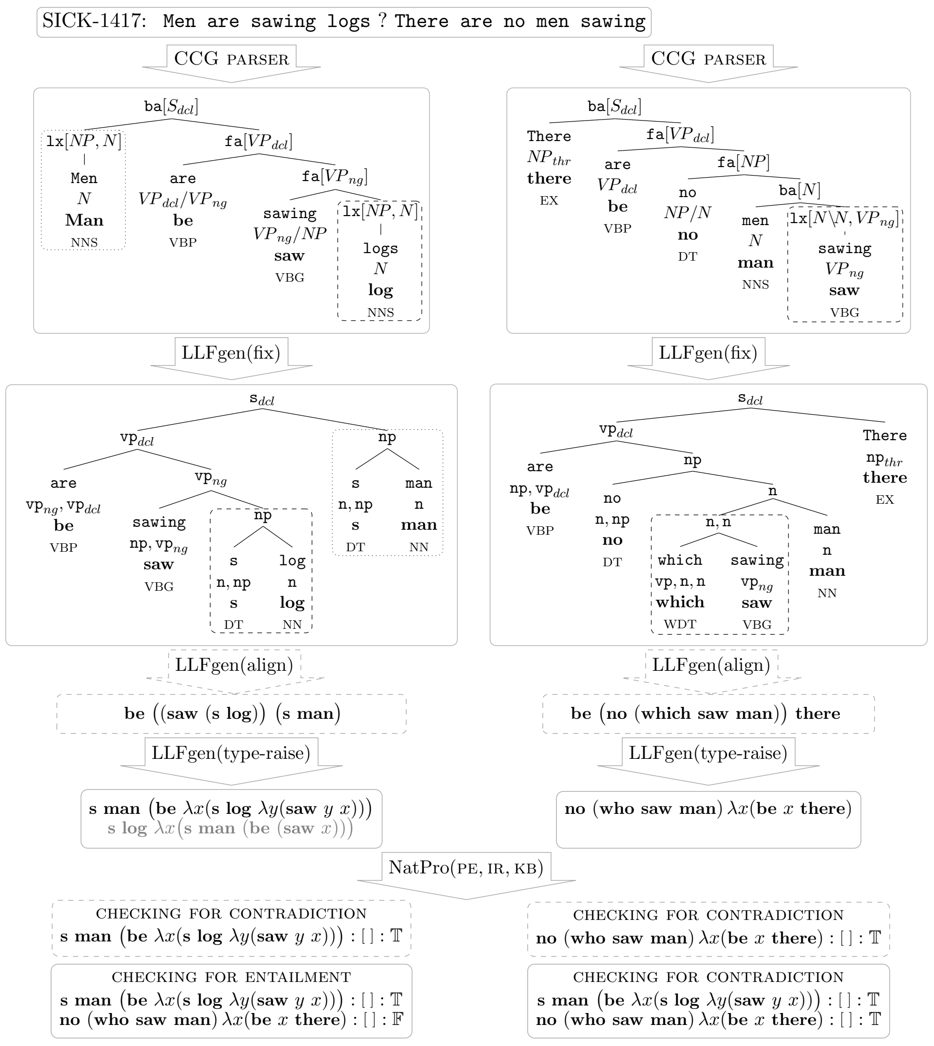
SICK-1417: Men are sawing logs ? There are no men sawing
CCG PARSER
LLFgen(fix)
LLFgen(align)
be ((saw (s log)) (s man))
be (no (which saw man)) there
LLFgen(type-raise)
s man (be λx(s log λy(saw y x)))
s log λx(s man (be (saw x)))
no (who saw man) λx(be x there)
NatPro(PE, IR, KB)
CHECKING FOR CONTRADICTION
CHECKING FOR ENTAILMENT
s man (be λx(s log λy(saw y x))) : [ ] : 𝕋
no (who saw man) λx(be x there) : [ ] : 𝔽
no (who saw man) λx(be x there) : [ ] : 𝕋

Figure 4. LangPro processes the TEP from SICK[21]. The CCG derivations obtained from a parser, namely C&C, are presented as a tree. The terminal nodes are annotated with a token, a CCG category, a lemma and a part of speech (POS) tag while the non-terminal ones are marked with a CCG category and a combinatory rule that combines the constituent(s). The constituents with a type changing rule and their fixed versions are framed. The term $\mathtt{s}_{\mathtt{n},\mathtt{vp},\mathtt{s}}$ stands for a plural quantifier. VP_i category and $\mathtt{vp}_i$ type abbreviate $S_i\backslash NP$ and $(\mathtt{np}, \mathtt{s}_i)$ respectively, where i is a category feature employed by the CCG parsers.

example, there are two LLFs for the premise due to the possible ambiguity between the quantifier scopes while for the conclusion only one LLF is generated. At this moment, LLFgen does not take the semantics of a quantifier into account during type-raising; that is why despite their semantic equivalence, both LLFs for the premise are generated. In the current settings of LangPro, for each CCG derivation only the first LLF is considered by the prover. Notice that the scope order in the first LLF usually resembles the order in the surface level.

NatPro first checks each LLF for contradiction.[9] If all LLFs are found self-consistent, they are checked for entailment, and if no proof is found then they are checked for contradiction (see the tableau in Fig. 5 which proves the running example as entailment). If the entailment relation is proved, there is no need for checking the LLFs for contradiction. In this way, neutral problems are the most time-consuming; for a neutral single-premised problem, in total 8 tableaux are constructed for the aligned and non-aligned LLFs.

4.2 Performance

For the training and evaluation purposes, we use the SICK [21] and FraCaS [11] RTE datasets for the following reasons.[10] The datasets contain relatively short sentences which is expected to increase correct analysis by CCG parsers. The FraCaS problems require no lexical knowledge while the SICK problems require only linguistic knowledge. The training process is not fully automatized: if LangPro misclassifies a problem, the process is debugged—either a new tableau rule or knowledge fact is introduced or LLFgen is further improved. More details about the training and development phases can be found in [1, Sec. 5].

In Table 2 both versions of the prover, ccLangPro and easyLangPro, are evaluated and compared to state-of-the-art results.[11] ccLangPro achieves

[9]If an LLF is found self-contradictory, i.e. the tableau initiated by it closes, there is a high chance that the source CCG derivation is erroneous [3, Sec. 4]. If one of the LLFs is self-contradictory, the prover aborts and returns the neutral answer. The same decision is made when one of the LLFs has a different type from the rest.

[10]FraCaS is a small set containing semantically challenging multi-premise problems. It is available at: `http://www-nlp.stanford.edu/~wcmac/downloads`. Currently few RTE systems are able to cope with the FraCaS problems. On the other hand, SICK is a large dataset intended for compositional distributional semantic models. It was used as a benchmark at the SemEval RTE challenge [20]: `http://alt.qcri.org/semeval2014/task1`. The TEPs in both datasets are human annotated with three labels: entailment, contradiction and neutral. We adopt the partition of SICK from the RTE challenge [20] and refer to these parts as SICK-trial, SICK-train and SICK-test.

[11]In case of FraCaS, the training and testing data are the same for LangPro; so comparison to the system that did not have a close look at the test problems should be

Table 2. Performance of the versions of LangPro on FraCaS and SICK

(a) Results on FraCaS's section 1 on generalize quantifiers involving both single- (one) and multi- (all) premise problems

Systems' Acc%	One (44)	All (74)
NatLog'07 [18]	84	-
NatLog'08 [19]	**98**	-
NaturalLI [4]	95	-
L&S'13 [17]	70/89	50/80
DCS [27]	79	80
HOL [14]	-	77
ccLangPro	95	**85**
easyLangPro	80	**81**
Baseline (major)	45	50

(b) Results on the test portion of SICK(4927). uniLangPro is a prover unifying judgments of ccLLF- and easyLLF-based provers.

Systems	Prec %	Rec %	Acc %
Illinois-LH	81.56	**81.87**	**84.57**
ECNU	84.37	74.37	83.64
UNAL-NLP	81.99	76.80	83.05
SemantiKLUE	85.40	69.63	82.32
Meaning Factory	93.63	60.64	81.59
ccLangPro	97.53	57.26	80.90
easyLangPro	97.63	57.83	81.15
uniLangPro	**97.67**	61.01	82.50
Baseline (major)	-	-	56.69

higher results on FraCaS than easyLangPro. This is explained by using ccLLFs during the entire training process, which made LLFgen to work better on ccLLFs. Despite fitting LLFgen to ccLLFs, easyLangPro still obtains high results on FraCaS and even beats ccLangPro on SICK-test. uniLangPro is a prover unifying both provers: it finds a proof iff one of the provers finds a proof. On SICK, the versions of the prover is compared to the top systems of the SemEval challenge [20]. Although LangPro shows a low recall on SICK-test, it obtains a competitive accuracy along with an almost prefect precision. Note that most of these errors are due to the noisy gold labels in the dataset. Several problems with noisy gold labels are discussed in the next section. Additional information about the comparison based on the SICK data is given in [1] and [3, Sec. 5].

We briefly compare our prover to two related systems: NatLog [19] and NutCracker [6]. LangPro improves over NatLog in terms of having full-fledged logic and proof system over the logical forms. As a result it can process more complex and multi-premised TEPs. Compared to NutCracker, our prover can reason over higher-order and monotone terms. NatPro is also specially tuned for natural reasoning in contrast to the off-the-shelf provers and model builders incorporated in NutCracker. Further details of the comparison is given in [1, Sec. 6]. The demo version of LangPro is available online.[12]

understood in terms of the expressive power of a system. For SICK the test data, SICK-test, was held out during training, so comparison to related systems can be made in terms of performance too.

[12]`http://lanthanum.uvt.nl/labziani/tableau`

5 Solving textual entailment problems

The section shows in details how LangPro processes various problems drawn from the SICK [21] and FraCaS [11] datasets. Each presented problem is accompanied with its dataset ID, the gold label and two judgments by the prover—one based on C&C and another on EasyCCG derivations. The selected set of examples intend to highlight the issues of natural language theorem proving and give the clues for further improvements.

5.1 True entailments and contradictions

First, we are going to discuss the positive (entailment or contradiction) problems that were correctly proved either by ccLangPro or by easyLangPro.

$\boxed{\text{SICK-1417}}$ GOLD: `cont.` LangPro(C&C/EasyCCG): `cont./neut.`

$\frac{\text{Men are sawing logs}}{\text{There are no men sawing}}$

SICK-1417: this is the problem from Fig. 4 in Section 4. Both parsers correctly analyze the premise while in case of the conclusion only EasyCCG makes mistake: "*men sawing*" is identified as a constituent $[\text{men}_{N/N}\ \text{saw}_N]_N$. This mistake is crucial for easyLLFs hence the proof over them is not found. Fortunately, correct C&C derivations result in semantically adequate ccLLFs. LangPro proves them as inconsistent in 10 rule applications; see the proof in Fig. 5.
Conclusion: while the proof is not found with EasyCCG due to a wrong derivation, the C&C derivations salvage the situation.

$\boxed{\text{SICK-8147}}$ GOLD: `ent.` LangPro(C&C/EasyCCG): `ent./ent.`

$\frac{\text{The girl [in blue]}_1 \text{ is chasing the base runner } \big[\text{with a number [on the jersey]}_3\big]_2}{\text{The girl [in blue]}_1 \text{ is chasing the player } \big[\text{with a number [on the jersey]}_3\big]_2}$

SICK-8147: the problem contains sentences each having three PPs, marked with brackets and indexed; this makes the sentences challenging for the parsers. Apart from the different analyses for each NP[13], the derivation trees from C&C and EasyCCG also differ in PP-attachments. For both sentences, C&C treats PP_3 as an argument of "*number*" while EasyCCG analyzes it as a modifier of "*a number*". In both sentences, EasyCCG wrongly but in a consistent way treats PP_2 as a VP modifier; PP_2 gets mixed analyses

[13]EasyCCG usually analyzes NPs with post-modifiers in the NP-S style, i.e. a determiner is grouped with a head earlier than post-modifiers: $\big[[\text{the girl}]_{NP}\ [\text{in blue}]_{NP\backslash NP}\big]_{NP}$. On the other hand, C&C trained on rebanked CCGbank [13] prefers the Nom-S analysis: $\big[\text{the}_{NP/N}\ [\text{girl}_N\ [\text{in blue}]_{N\backslash N}]_N\big]_{NP}$.

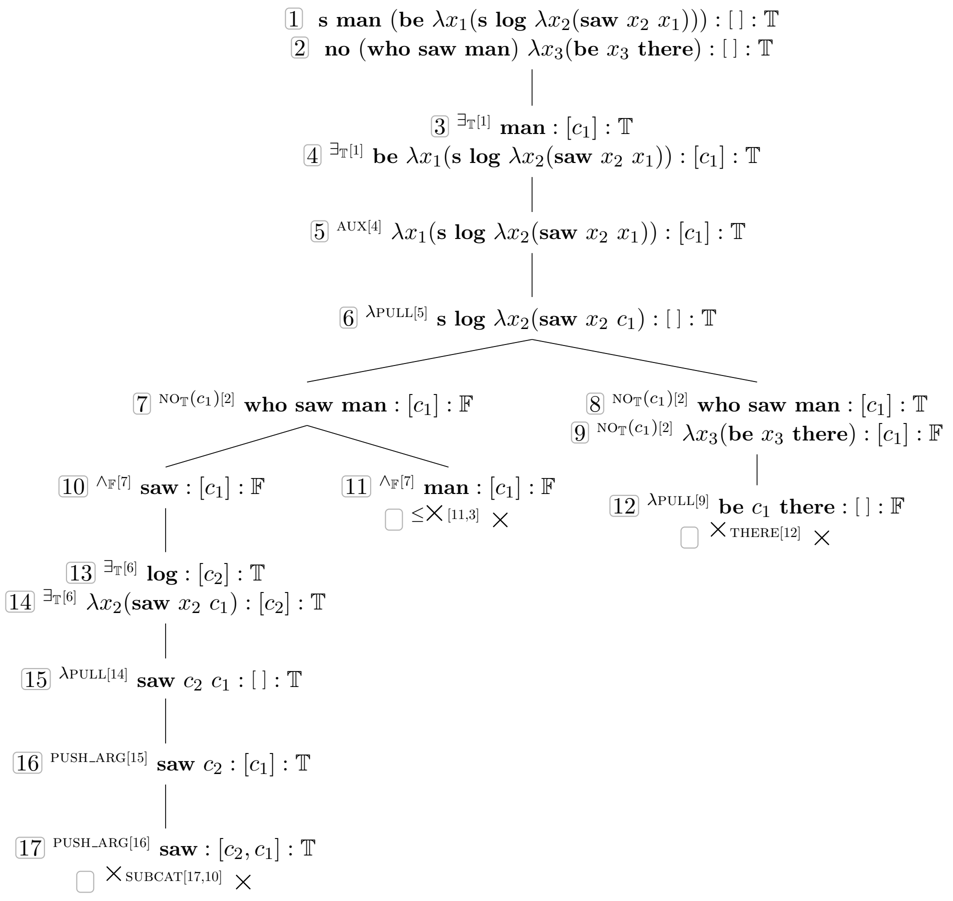
1 s man (be λx1(s log λx2(saw x2 x1))) : [ ] : 𝕋
2 no (who saw man) λx3(be x3 there) : [ ] : 𝕋
3 ∃𝕋[1] man : [c1] : 𝕋
4 ∃𝕋[1] be λx1(s log λx2(saw x2 x1)) : [c1] : 𝕋
5 AUX[4] λx1(s log λx2(saw x2 x1)) : [c1] : 𝕋
6 λPULL[5] s log λx2(saw x2 c1) : [ ] : 𝕋
7 NO𝕋(c1)[2] who saw man : [c1] : 𝔽
8 NO𝕋(c1)[2] who saw man : [c1] : 𝕋
9 NO𝕋(c1)[2] λx3(be x3 there) : [c1] : 𝔽
10 ∧𝔽[7] saw : [c1] : 𝔽
11 ∧𝔽[7] man : [c1] : 𝔽
≤× [11,3] ×
12 λPULL[9] be c1 there : [ ] : 𝔽
×THERE[12] ×
13 ∃𝕋[6] log : [c2] : 𝕋
14 ∃𝕋[6] λx2(saw x2 c1) : [c2] : 𝕋
15 λPULL[14] saw c2 c1 : [ ] : 𝕋
16 PUSH_ARG[15] saw c2 : [c1] : 𝕋
17 PUSH_ARG[16] saw : [c2, c1] : 𝕋
×SUBCAT[17,10] ×

Figure 5. The closed tableau proves SICK-1417 as entailment. The intuition behind the employed rules (e.g., λPULL, AUX, NO$_{\mathbb{T}}$, etc.) can be read from the tableau.

from C&C: correctly identified as a modifier of "*player*" in the conclusion but analyzed wrongly in the premise, like in case of EasyCCG. Due to these differences, the corresponding ccLLFs and easyLLFs, including their aligned versions, also differ from each other. In particular, the terms for PP$_2$ are aligned in the $\overline{\text{easy}}$LLFs; but for the $\overline{\text{cc}}$LLFs, the shorter subterms of "*a number on the jersey*" are aligned because C&C analyzes PP$_2$ in a mixed way and assigns different categories to "*with*" in the sentences.

For both versions of aligned LLFs, LangPro finds proofs for the entailment relation. Due to the poor alignment for the $\overline{\text{cc}}$LLFs, the tableau was closed in 20 rule applications while for the $\overline{\text{easy}}$LLFs, with the better alignment, in 8 applications. Despite the wrong attachments of PP$_2$ in the

$\overline{\text{easy}}$LLFs, proof search is more efficient because the attachments were *consistent* which contributed to the better alignment. In case of the $\overline{\text{cc}}$LLFs, the attachments of PP$_2$ were inconsistent (though one of them was correct) which finally costs much in terms of a lengthy proof. Moreover, finding a proof for the $\overline{\text{cc}}$LLFs would be impossible if we did not introduce the $\times$PP_ATT$_{\mathbb{T}}$ rule, which abstracts from inconsistency in PP-attachments. As a result, the rule identifies the nodes in (3) as inconsistent; terms written in CamelCase are the aligned subterms while the individual constant c stands for "*the base runner*".

Capturing the entailment **base runner** $\leq$ **player** is a main part of solving SICK-8147. This piece of information is necessary for the tableau to be close. The multi-sense approach tries to find some senses of the nouns for which the entailment relation holds. Such senses are found in the knowledge base as the synset {"*base runner*", "*runner*"} is indirectly subsumed by {"*player*", "*participant*"} synset in WordNet.[14]

$$\frac{\begin{array}{r}\{\textbf{with aNumberOnTheJersey}\} : \textbf{chase} : [c, \textbf{theGirlInBlue}] : \mathbb{T}\\ \textbf{with} : [\textbf{aNumberOnTheJersey}, c] : \mathbb{F}\end{array}}{\times}\ \times\text{PP\_ATT}^{*}_{\mathbb{T}} \qquad (3)$$

Conclusion: the consistent (possibly wrong) analyses of PP-attachments leads to the better alignment of terms, which itself contributes to the shorter proof. A wrongly attached structurally ambiguous PP can be identified and correctly interpreted with the help of the special rules. The multi-sense approach also works well for this case.

$\boxed{\text{FraCaS-18}}$ GOLD: `ent.` LangPro(C&C/EasyCCG): `neut./ent.`

Every European has the right $\big[[\text{to live in Europe}]_{VP_{to}}\big]_{N\backslash N}$
Every European is a person
Every person who has the right [to live in Europe] can travel freely within Europe
Every European can travel freely within Europe

FraCaS-18: the textual entailment contains multiple premises but this is not a problem for the prover. The challenge in this example is to obtain decent derivations and to convert them in LLFs. The C&C line fails in the beginning when the parser fails to return the derivations for the first two sentences which contain relevant information for the entailment. Fortunately, EasyCCG gets all the sentences parsed. The produced easyLLFs are

[14]Tthe prover is also able to derive **base runner** $\leq$ **player** relation by first capturing **base runner** $\leq$ **runner** using a rule for subsective adjectives and then combining it with **runner** $\leq$ **player** relation retrieved from WordNet. Based on the information **runner** $\leq$ **player**, unfortunately the multi-sense approach also leads to the proof of "*A runner won*" entails "*A player won*".

not proper λ-terms since LLFgen, at this moment, does not have a remedy for the rule that changes the syntactic type $\mathtt{vp}_{to}$ into $(\mathtt{n}, \mathtt{n})$. Despite this shortcoming, LangPro is still able to operate on the easyLLFs and find a proof for entailment in 43 rule applications (actually, only a few of those applications contribute to the proof). In case the proof needed to decompose the LLF corresponding to "*the right ... Europe*", due to the unexplained type-changing rule, the prover would fail to do so and fail to find a proof. Notice that in this example, the aligner is useless as there is no multiword phrase shared by all the sentences.
Conclusion: C&C failed to parse several sentences, however EasyCCG saved the situation. The obtained easyLLFs were not well-formed terms, nevertheless the prover is able to process them as long as the decomposition of the ill-formed subterms is not necessary for the proof.

$\boxed{\text{SICK-1207}}$ GOLD: `cont.` LangPro(C&C/EasyCCG): `cont./cont.`

A woman is not talking on a telephone
A woman is talking on a telephone

SICK-1207: this is a dubious case. One might identify the problem as neutral or contradiction depending whether the indefinite NPs "*a woman*" and "*a telephone*" are co-referencing to the same referents. The human annotations in SICK show a strong tendency towards co-reference of this kind of NPs (the similar problems, SICK-363 and 1989, with the similar judgments are given in Table 3).

In order to support the co-reference, we choose a simple solution that comes for free with the current settings of LangPro. In particular, the aligner is used for this purpose: aligning the identical indefinites make them to refer to the same entity. The aligned LLFs for the problem are given in (4) and (5). Using the aligned LLFs, the prover is able to find a proof in 3 rule applications.

$$\mathbf{not}_{\mathrm{vp,vp}}\ \mathbf{beTalkOnATelephone}_{\mathrm{vp}}\ \mathbf{aWoman}_{\mathrm{np}} \tag{4}$$

$$\mathbf{beTalkOnATelephone}_{\mathrm{vp}}\ \mathbf{aWoman}_{\mathrm{np}} \tag{5}$$

Conclusion: the simple solution with the alignment technique accounts enough well for the co-reference of indefinite NPs. It also makes proofs extremely short.[15]

Above we tried to give the TEPs that were correctly classified in spite of the various shortcomings. Usually this kind of TEPs are rare. For exam-

[15]If in the premise "*a woman*" is replaced by "*a person*", the alignment approach cannot contribute to the proof. More general solution to the co-reference of indefinites is achieved when the negation takes a wide scope. Implementing the latter approach requires further development of LLFgen. This approach is left for future work.

ple, while in our examples the judgments of ccLangPro and easyLangPro diverged in the half of the cases, based on the development set—SICK-train (4500 problems)—the judgments diverge only for 3.3% of the TEPs.

5.2 False entailments and contradictions

Given the almost perfect precision of the prover, the false positive problems represents a special case of interest.

$\boxed{\text{SICK-8461}}$ GOLD: `neut.` LangPro(C&C/EasyCCG): `cont./cont.`

A man with no hat [is sitting on the ground]$_1$

A man with a backwards hat [is sitting on the ground]$_1$

SICK-8461: the ccLLFs and easyLLFs for both sentences are quite similar; the ccLLF and easyLLF for the premise are given in (6) and (7), respectively. The only difference is in the analyses of the verb phrases: whether **be** takes **sit** or the whole verb phrase as an argument. This difference has no influence on proof search since the auxiliaries are currently treated as the identity function.

$$\textbf{no hat}\ \lambda y\big(\textbf{a}\ \big(\textbf{with}\ y\ \textbf{man}\big)\ \lambda x\big(\textbf{the ground}\ \lambda z(\textbf{on}\ z\ (\textbf{be sit})\ x)\big)\big) \tag{6}$$

$$\textbf{no hat}\ \lambda y\big(\textbf{a}\ \big(\textbf{with}\ y\ \textbf{man}\big)\big(\textbf{be}\ \lambda x(\textbf{the ground}\ \lambda z(\textbf{on}\ z\ \textbf{sit}\ x))\big)\big) \tag{7}$$

$$\texttt{a}_{(et)(et)t}\ \lambda y\big(\texttt{no}_{(et)(et)t}\ \texttt{hat}_{et}\ \lambda x(\texttt{with}_{e(et)et}\ x_e\ \texttt{man}_{et}\ y_e)\big)\ \texttt{sleep}_{et} \tag{8}$$

In contrast to the surface level, **no hat** takes the widest scope in the LLFs. The reason is usage of the terms with syntactic types. While using the terms of semantic types, it is possible that **no hat** takes a narrow scope, see (8). But in case of the syntactic types, **no hat** can not be type-raised in the PP because a determiner of type $(\texttt{n}, (\texttt{np}, \texttt{s}), \texttt{s})$ cannot take a term of type $(\texttt{np}, t)$ or (e, t) for its second argument.[16]

It is obvious that if the sentences were understood with "*no hat*" and "*a backwards hat*" having the widest scope, then they would be inconsistent. This is why the prover classifies the problem as contradiction. The proofs for both versions of LLFs were found in 5 rule applications. The alignment of VP$_1$ does not affects the proof search as the relevant terms "*no hat*" and "*a backwards hat*" are analyzed and contrasted to each other before VP$_1$ is processed.

Conclusion: the desirable scope order for quantifiers is not obtained due to less-flexible syntactic types, which in the end leads to the wrong prediction. This mistake seems minor taking into account that the similar

[16]This issue can be solved by introducing a semantic counterpart of the determiner that is of type $(et)(et)t$, but this itself will further require introduction of semantic counterparts of other terms. The latter complicates the proof search.

problems, e.g., SICK-8562 in Table 3, receive mixed judgments (neutral or contradiction) in SICK.

$\boxed{\text{SICK-7402}}$ GOLD: `neut.` LangPro(C&C/EasyCCG): `cont./cont.`

There is [[no man] and [child kayaking through gentle waters]]

A man and a young boy are riding in a yellow kayak

SICK-7402: the problem is neutral as the sentences are informative with respect to each other, but the prover identifies it as contradiction. The reason is the wrong derivation trees where "*no*" scopes only over "*man*". In this way, the premise implies that there is no man while the conclusion asserts the contrary—a man is riding in a kayak. The prover identifies this inconsistency and classifies the problem as contradiction.
Conclusion: the mistakes by the C&C and EasyCCG parsers misled the prover. In general, it is very rare that the mistake by a parser leads to a false positive.

$\boxed{\text{FraCaS-58}}$ GOLD: `neut.` LangPro(C&C/EasyCCG): `neut./ent.`

Most Europeans [who are resident in Europe] can travel freely within Europe

Most Europeans can travel freely within Europe

FraCaS-58: the judgments based on the ccLLFs and easyLLFs differ from each other. The rationale for the proof found over the easyLLFs is the non-restrictive (i.e. appositional) interpretation of the relative clause. Since the EasyCCG derivations were not used during development of LLFgen, it failed to correct the EasyCCG derivation for the premise.
Conclusion: LLFgen could not correct an unobserved mistake in the EasyCCG derivation. As a result, the mistake caused the false proof for entailment.

$\boxed{\text{SICK-5264}}$ GOLD: `neut.` LangPro(C&C/EasyCCG): `ent./ent.`

A person is folding a sheet

A person is folding a piece [of paper]$_1$

SICK-5264: both decisions of the prover are false according to the gold label. Different analyses of PP$_1$—as a noun argument by C&C and as an NP modifier by EasyCCG—do not affect the final judgments because all the argument PPs are also treated as modifier PPs with the help of the rules.

The reason for the contradiction proofs is that "*a sheet*" is "*a piece of paper*" according to the multi-sense approach: in WordNet "*sheet*" has a sense that is a hyponym of a sense of "*paper*", and there is the tableau rule that identifies "*a piece of paper*" as "*paper*". The proofs are found in 18 rule appreciations.

Table 3. The false positive examples and the problems with noisy gold (G) labels. The problems are drawn from SICK. The words that were related to each other by LangPro (LP) are in bold while the unrelated ones in italic.

ID	G/LP	Premise	Conclusion
1405	N/E	A **prawn** is being cut by a woman	A woman is cutting **shrimps**
1481	N/E	A deer is jumping over a **wall**	The deer is jumping over the **fence**
1777	N/E	A **boy** is happily playing the piano	A piano is being played by a **man**
4443	N/E	A man is singing to a **girl**	A man is singing to a **woman**
2870	N/C	Two people are riding a **motorcycle**	Nobody is riding a **bike**
2868	E/N	Two people are *stopping* on a **motorcycle**	Two people are *riding* a **bike**
6258	E/N	A *policeman* is sitting on a **motorcycle**	The cop is sitting on a *police* **bike**
344	N/C	P: An Asian woman in a crowd is not carrying a black bag C: An Asian woman in a crowd is carrying a black bag	
545	N/C	P: A woman is standing and is not looking at the waterfall C: A woman is sitting and looking at the waterfall	
8913	N/C	A couple is not looking at a map	A couple is looking at a map
363	C/C	A soccer ball is not rolling into a goal net	A soccer ball is rolling into a goal net
1989	C/C	A girl is playing the guitar	A girl is not playing the guitar
8562	C/N	P: A man *in a hat* is standing outside of a green jeep C: A man *with no hat* is standing outside of a green jeep	

Conclusion: the multi-sense approach makes a co-reference that leads the prover to find a proof for contradiction.

The other false positives that were proved in the same vein as SICK-5264 are give in the upper part of Table 3. The problems were proved due to the relations, like **wall** $\leq$ **fence** and **girl** $\leq$ **woman**, licensed by the multi-sense approach. Notice noise with respect to **motorcycle** $\leq$ **bike** relation. While SICK-2870 rejects it, SICK-2868 and 6258 presuppose the relation. Unfortunately, our prover was not able to capture the latter two entailments as it failed to relate other lexical entries.

On the SICK dataset, the prover rarely finds false proofs and when it does, the multi-sense approach or the noisy labels are the reason in around 80% of the cases. ccLangPro has no false proofs on the first section of FraCaS. On the other hand, easyLangPro finds two false proofs due to non-restrictive relative clauses (e.g., FraCaS-58) and one due to a wrong analysis of the expression "*at most*".

5.3 False neutrals

There can be several reasons for a false neutral: starting from the mistakes by the CCG parsers finishing with a poor strategy for proof search. The prover shows a large number of false neutrals on SICK. In order to find out the reason behind it, we randomly draw 200 problems from SICK-train and analyzed the false positives found there. Around a half of the false positives

Table 4. Examples of false neutrals from SICK. The factors for the failure (Fail) are noisy gold labels (G), the mistakes by the parsers (P), a lack of rule (R) and a lack of knowledge (K). Each problem is marked with the reason of failure.

ID	G	Fail	Premise	Conclusion
4720	E	G	A *monkey* is practicing martial arts	A *chimp* is practicing martial arts
4275	E	R	*A man and a woman* are shaking hands	*Two persons* are shaking hands
4553	E	R	P: A man is emptying a *container made of plastic* C: A man is emptying a *plastic container*	
2763	C	K	A man and woman *are talking*	A man and a woman *are silent*
4974	C	K	Someone is holding a *hedgehog*	Someone is holding a *small animal*
6447	C	P	P: [A small boy [in a yellow shirt]] is laughing on the beach C: There is no small boy [in a yellow shirt [laughing on the beach]]	

were due to knowledge sparsity (see some of the examples in Table 4). A lack of the tableau rule was a reason for a quarter of the problems. This kind of problems also include the cases where an absent paraphrase can be captured with a schema, e.g., X *made of* $Y \to YX$, like in SICK-4553. The entailments concerning the cardinality need assist from the tableau rules too. The rest of the false neutrals are evenly provoked by noisy gold labels and the mistakes coming from the parsers.

6 Conclusion

We presented the tableau-based theorem prover for natural language, called LangPro. The prover is able to reason over wide-coverage natural language text with the help of the CCG parsers and the module LLFgen producing the logical forms. After training on the SICK and FraCaS dataset, LangPro achieves competitive results with respect to the state-of-the-art RTE systems. The noteworthy virtues of the prover are (i) the almost perfect precision (despite the noisy gold labels of SICK, nearly 98% of the proofs are correct), (ii) the expressive higher-order logic with *natural-looking* formulas, (iii) the proof search strategy specially suited for natural reasoning and (iv) the explanatory decision procedure based on the analytic tableau method. On the other hand, the prover is a rule-based system with a pipeline architecture, which makes it brittle and expensive for training. To the best of our knowledge, LangPro is the only wide-coverage RTE system that is based on natural logic and is able to reason over multiple premises.

For each pair of the human and prover judgments, several textual entailment problems were discussed in details. Due to the accurate nature of LangPro, special attention was paid to the false predictions. For SICK problems, knowledge sparsity was identified as one of the key factors for the relatively low recall.

In future work, we plan to address the observed problematic issues: knowledge acquisition, word sense disambiguation and generation of semantically adequate LLFs. We intend to populate the knowledge base with further relations from WordNet (e.g., similarity for adjectives and entailment and causation for verbs) and to explore other lexical or phrasal databases for future integration. Both SICK and FraCaS data contain relatively short sentences. It would be interesting to test the prover and the LLF generator module against more naturally occurring text, for instance, the RTE datasets collected from newswire text. The quality of LLFs can be improved by exploring the n-best derivations of EasyCCG.

References

1. Abzianidze, L. A Tableau Prover for Natural Logic and Language. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.2492–2502. ACL (2015)

2. Abzianidze, L. Towards a Wide-coverage Tableau Method for Natural Logic. In: Murata, T., Mineshima, K., Bekki, D. (eds.), *New Frontiers in Artificial Intelligence*, LNCS, vol. 9067, pp. 66–82. Springer Verlag (2015)

3. Abzianidze, L. A Pure Logic-Based Approach to Natural Reasoning. In Brochhagen, Th., Roelofsen, F., Theiler, N. (Eds.) *Proceedings of the 20th Amsterdam Colloquium*, pp 40-49. ILLC, University of Amsterdam (2015)

4. Angeli, G., Manning, C.D. NaturalLI: Natural Logic Inference for Common Sense Reasoning. *Proceedings of Empirical Methods in Natural Language Processing (EMNLP).* pp. 534–545, ACL (2014)

5. van Benthem, J.F.A.K. Essays in Logical Semantics. Reidel, Dordrecht (1986)

6. Bos, J., Markert, K. Recognising textual entailment with logical inference. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 628-–35. ACL (2005)

7. Clark, S., Curran, J.R. Wide-Coverage Efficient Statistical Parsing with CCG and Log-linear Models. *Computational Linguistics*, 33(4) (2007)

8. Dagan, I., Glickman, O., Magnini, B. The PASCAL recognizing textual entailment challenge. In Quioñero-Candela et al. (eds.),

First PASCAL Machine Learning Challenges Workshop, pp. 177-–190. Springer-Verlag (2006).

9. Dagan, I., Roth, D., Sammons, M., Zanzotto, F.M. Recognizing Textual Entailment: Models and Applications. *Morgan and Claypool Publishers* (2013)

10. D'Agostino, M., Gabbay, D.M., H ahnle, Posegga, J. (eds.) Handbook of Tableau Methods. Springer (1999)

11. Cooper, R., Crouch, D., van Eijck, J., Fox, C., van Genabith, J., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M., Pulman, S. Using the Framework. *Technical Report LRE 62-051 D-16.* The FraCaS Consortium (1996)

12. Fellbaum, Ch. (eds.): WordNet: an Electronic Lexical Database. *MIT press* (1998)

13. Honnibal, M., Curran, J.R., Bos, J. Rebanking CCGbank for Improved NP Interpretation. In *Proceedings of the 48th ACL Conference*, pp. 207–215 (2010)

14. Mineshima, K., Martínez-Gómez, P., Miyao, Y., Bekki, D. Higher-order Logical Inference with Compositional Semantics. In *Proceedings of the 48th ACL Conference*, pp. 2055–2061, ACL (2015)

15. Lakoff, G. Linguistics and Natural Logic. In Davidson, D., Harman, G. (eds.) *Semantics of Natural Language.* Synthese Library, vol. 40, pp. 545–665, Springer (1972)

16. Lewis, M., Steedman, M. A* CCG Parsing with a Supertag-Factored Model. In *Proceedings of the EMNLP 2014*, pp. 990-–1000. ACL (2014)

17. Lewis, M., Steedman, M. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics (TACL)*, vol. 1, pp. 179–192. ACL (2013)

18. MacCartney, B., Manning, C. D. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 193-–200. ACL (2007)

19. MacCartney, B., Manning, C. D. Modeling Semantic Containment and Exclusion in Natural Language Inference. In *Proceedings of Coling-08*, Manchester, UK (2008)

20. Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. *Proceedings of the 8th SemEval* (2014).

21. Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R. A SICK Cure for the Evaluation of Compositional Distributional Semantic Models. In *Proceedings of LREC*, Reykjavik (2014)

22. Muskens, R. An Analytic Tableau System for Natural Logic. In: Aloni, M., Bastiaanse, H., de Jager, T., Schulz, K. (eds.) *Logic, Language and Meaning.* LNCS, vol. 6042, pp. 104–113. Springer, Heidelberg (2010)

23. Navigli, R., Litkowski, K. C., Hargraves, O. Semeval-2007 Task 07: Coarse-Grained English All-Words Task. In: *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval)*, pp. 30–35. Prague, (2007)

24. Snyder, B., Palmer, M. The English All-Words Task. In: *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SensEval-3)*, pp. 41–43. Barcelona (2004)

25. Steedman, M. The Syntactic Process. *MIT press* (2000)

26. Sánchez Valencia, V. Studies on Natural Logic and Categorial Grammar. *PhD dissertation*, University of Amsterdam (1991)

27. Tian, R., Miyao, Y., Matsuzaki, T. Logical Inference on Dependency-based Compositional Semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 79–89, ACL (2014)