An Analytic Tableau System for Natural Logic

Reinhard Muskens

Tilburg Center for Logic and Philosophy of Science
r.a.muskens@uvt.nl
http://let.uvt.nl/general/people/rmuskens/

Abstract. In this paper we develop the beginnings of a tableau system for natural logic, the logic that is present in ordinary language and that us used in ordinary reasoning. The system is based on certain terms of the typed lambda calculus that can go proxy for linguistic forms and which we call Lambda Logical Forms. It is argued that proof-theoretic methods like the present one should complement the more traditional model-theoretic methods used in the computational study of natural language meaning.

1 Introduction

Logic has its roots in the study of valid argument, but while traditional logicians worked with natural language directly, modern approaches first translate natural arguments into an artificial language. The reason for this step is that some artificial languages now have very well developed inferential systems. There is no doubt that this is a great advantage in general, but for the study of natural reasoning it is a drawback that the original linguistic forms get lost in translation. An alternative approach would be to develop a general theory of the natural logic behind human reasoning and human information processing by studying formal logics that operate directly on linguistic representations. That this is possible we will try to make plausible in this paper. It will turn out that one level of representation, that of Logical Form, can meaningfully be identified with the language of an existing and well-understood logic, a restricted form of the theory of types. It is not difficult to devise inference systems for this language, and it is thus possible to study reasoning systems that are based directly on language.

We will define a tableau system and will place in focus tableau rules that are connected with certain properties of operators that seem important from a linguistic point of view. Our aim will not so much be to provide a proof system that is complete with respect to the semantics of our representations, but to provide rules that can be argued to be natural. The paper's purpose, therefore, is to contribute to the field of *natural logic*.[1]

[1] Early contributions to natural logic are [16] and [22]. The research line we base ourselves upon is exemplified in [10, 11, 3, 4, 21, 9, 5, 12, 24, 17, 18].

2 Lambda Logical Forms

For our purpose it will be of help to have representations of natural language expressions that are adequate both from a linguistic and from a logical point of view. At first blush, this may seem problematic, as it may be felt that linguistics and logic require completely different and competing properties from the representations they use, but in fact the typed lambda calculus provides what we need, or at least a good approximation to it. In order to obtain a class of terms with linguistic relevance we will restrict attention to those (simply typed) lambda terms that are built up from variables and *non-logical* constants, with the help of application and lambda abstraction and will delimit this class further by the restriction that only variables of individual type are abstracted over. The resulting terms, which will be called *Lambda Logical Forms* (LLFs), are often very close to linguistic expressions, as the following examples illustrate.

(1) a. `((a woman)walk)`
    b. `((if((a woman)walk))((no man)talk))`
    c. `(mary(think((if((a woman)walk))((no man)talk))))`
    d. `((a woman)(`λx`(mary(think((if(walk`x`))((no man)talk))))))`
    e. `(few man)`λx`.(most woman)`λy`.like`xy

The terms in (1) were built up in the usual way, but no *logical* constants, such as $=, \forall, \exists, \rightarrow, \wedge, \vee, \neg$ and the like, were used in their composition. The next section will make a connection between some of the non-logical constants used in (1) and logical ones, but this connection will take us from natural representations of linguistic expressions to rather artificial ones. Lambda terms containing no logical constants will therefore continue to have a special status.

Lambda Logical Forms come close to the Logical Forms that are studied in generative grammar. For example, in [13] trees such as the one in (2a) are found, strikingly similar to the λ-term in (2c).

(2) a. $[_{\mathrm{S}}[_{\mathrm{DP}}$ every linguist$][1[_{\mathrm{S}}$ John$[_{\mathrm{VP}}$ offended $t_1]]]]$
    b. `((every linguist)(`λx_1`(john(offend `x_1`))))`

3 A Natural Logic Tableau System

In this section we will discuss a series of rules for a tableau system directly based on LLFs. While tableau systems usually only have a handful of rules (roughly two for each logical operator under consideration), this system will be an exception. There will be many rules, many of them connected with special classes of expressions. Defining a system that comes even close to adequately describing what goes on in ordinary language will be a task far greater than what can be accomplished in a single paper and we must therefore contend ourselves with giving examples of rules that seem interesting. Further work should lead to less incomplete descriptions. Since the rules we consider typically are connected

to some algebraic property or other (such as monotonicity or anti-additivity—see below), it will also be necessary to specify to which class of expressions each rule applies. Describing exactly, for example, which expressions are monotone increasing in any given language requires a lot of careful linguistic work and for the moment we will be satisfied with providing examples (here: `some`, `some N`, `every N`, `many N`, and `most N`).

Familiarity with the method of tableaus will be assumed.

3.1 Tableau Entries

We will work with signed tableaus in which entries are formed by the rule that if A is an LLF of type[2] $\langle\vec{\alpha}\rangle$ and $\vec{C}$ is a sequence of constants or LLFs of types $\vec{\alpha}$, then $T\vec{C} : A$ and $F\vec{C} : A$ are tableau entries.

An entry $T\vec{C} : A$ ($F\vec{C} : A$) intuitively states that $A\vec{C}$ is true (false).

3.2 Closure

It will be assumed that the lexicon provides us with certain primitive entailment relations, such as `lark` $\leq$ `bird` and `no` $\leq$ `few`. A tableau branch is *closed* if it either contains both $T\vec{C} : A$ and $F\vec{C} : A$ or contains $T\vec{C} : A$ and $F\vec{C} : B$, where $A \leq B$ is lexical knowledge in this way.

A tableau is closed if all its branches are closed.

3.3 Rules Deriving from the Format

The format we have chosen itself validates some rules. First, we are only interested in LLFs up to $\beta\eta$ equivalence and lambda conversions can be performed at will. Second, the $X\vec{C} : A$ format (where X is T or F) validates the following rules.

(3)
$$\begin{array}{c} X\vec{C} : AB \\ \downarrow \\ XB\vec{C} : A \end{array} \qquad\qquad \begin{array}{c} XB\vec{C} : A \\ | \\ X\vec{C} : AB \end{array}$$

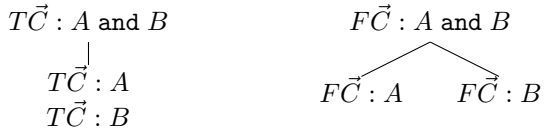
So we can shift arguments to the front and shift them back again.

3.4 Boolean Rules

We can now give rules for the operators `and`, `or` and `not`, the first two of which we write between their arguments, that are much like the rules for $\wedge$, $\vee$ and $\neg$ in signed tableau calculi. What is different here is that these rules are given for conjunction, disjunction and complementation in all categories, not just the category of sentences.

[2] Types will be relational, as in [20]. A relational type $\langle\alpha_1 \ldots \alpha_n\rangle$ is equivalent to the functional type $\alpha_1 \to \cdots \to \alpha_n \to t$ and $\langle\rangle$ is equivalent to t.

(4) a.

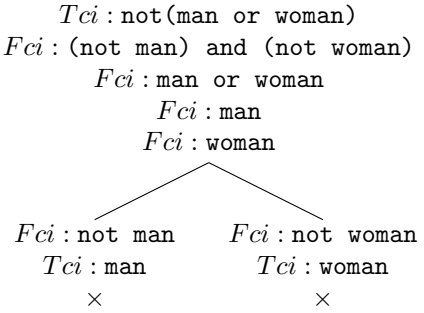


b.

c.

Here is a tableau showing that `not(man or woman)` entails `(not man) and (not woman)`.

(5)



In order to refute the possibility that some object c and some world i satisfy `not(man or woman)` but do not satisfy `(not man) and (not woman)` a tableau was developed which starts from the counterexample set

$$\{Tci : \texttt{not(man or woman)},\ Fci : \texttt{(not man) and (not woman)}\}\ .$$

Since the tableau closes (× signals branch closure) the possibility is indeed refuted.

While `and`, `or` and `not` seem to be operative in all categories, `if` is sentential. We formulate its rules as follows. Note that sentences still need a parameter (here: i) since their type is $\langle s \rangle$, not just $\langle\rangle$.

(6)

3.5 Rules for Monotonic Operators

The rules we have discussed until now were either completely general or operated on specific words (constants), but it has been observed that natural reasoning

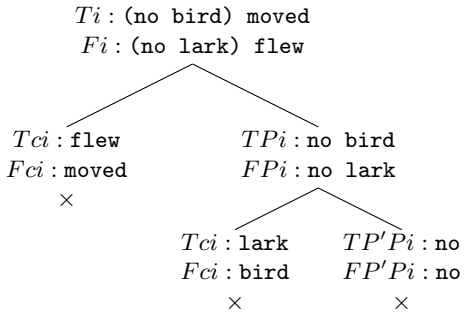


Table 1. Tableau for *no bird moved; therefore no lark flew*

hinges on properties that attach to certain *groups* of expressions. Let us write $\subset_i$ for the relation that obtains between relations M and M' of the same type $\langle\vec{\gamma}s\rangle$ if $(\lambda\vec{x}.M\vec{x}i) \subset (\lambda\vec{x}.M'\vec{x}i)$. A relation G of type $\langle\langle\vec{\alpha}s\rangle\vec{\beta}s\rangle$ is called *upward monotone* if $\forall XY\forall i(X \subset_i Y \rightarrow GX \subset_i GY)$ (where X and Y are of type $\langle\vec{\alpha}s\rangle$). Examples of upward monotone expressions (already mentioned above) are `some`, `some N`, `every N`, `many N`, `most N` (where `N` varies over expressions of type $\langle es\rangle$), but also `Mary`. Here is a tableau rule for upward monotone (mon↑) expressions.

(7)
$$\begin{array}{l} T\vec{C}i : GA \\ F\vec{C}i : HB \end{array}$$
where $\vec{c}$ and b are fresh, provided G or H is mon↑

$$\begin{array}{ll} T\vec{c}i : A & Tb\vec{C}i : G \\ F\vec{c}i : B & Fb\vec{C}i : H \end{array}$$

And here is a dual rule for expressions that are downward monotone, i.e. that satisfy the property $\forall XY\forall i(X \subset_i Y \rightarrow GY \subset_i GX)$. Examples are `no`, `no N`, `every`, `few`, and `few N`.

(8)
$$\begin{array}{l} T\vec{C}i : GA \\ F\vec{C}i : HB \end{array}$$
where $\vec{c}$ and b are fresh, provided G or H is mon↓

$$\begin{array}{ll} T\vec{c}i : B & Tb\vec{C}i : G \\ F\vec{c}i : A & Fb\vec{C}i : H \end{array}$$

Using the second of these rules, the tableau in Table 1 shows, by way of example, that `no bird moved` entails `no lark flew`. Table 2 gives a more complex example, showing that *each person who Mary touched ran* entails *most students who Mary kissed moved*. Here the rules employed for `who` are essentially those for `and`.

3.6 Other Rules Connected to Algebraic Properties

Upward and downward monotonicity are not the only algebraic properties that seem to play a pivotal role in language. There is a literature starting with [25]

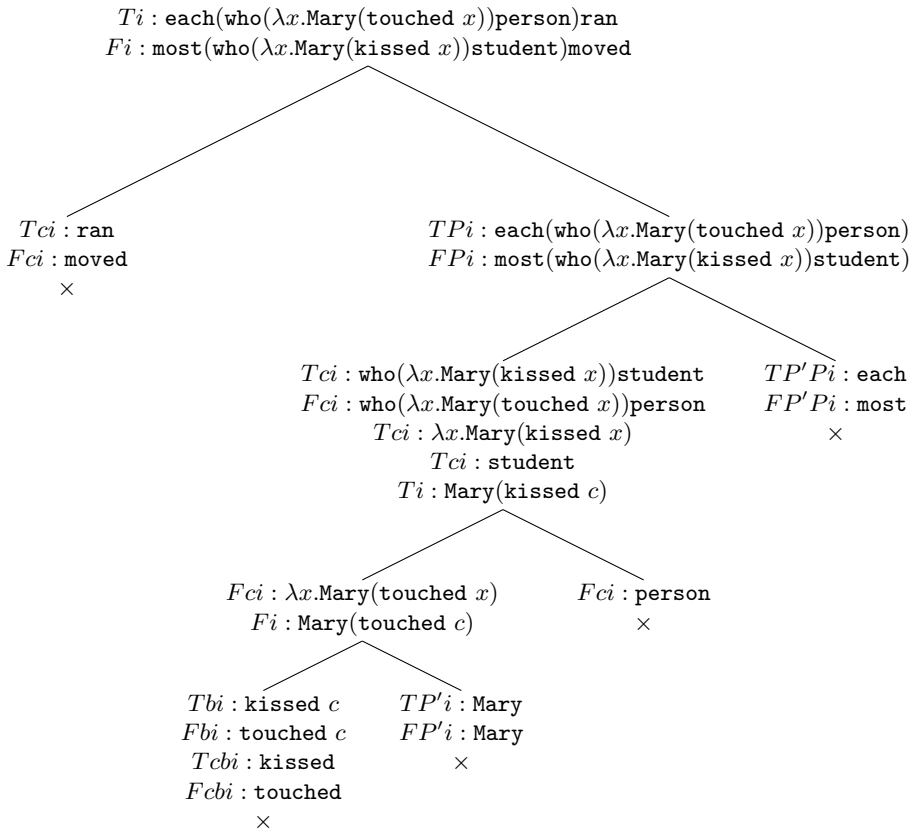
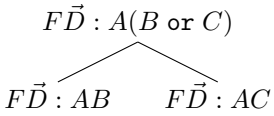


Table 2. Tableau for *each person who Mary touched ran; therefore most students who Mary kissed moved*

singling out *anti-additivity* as linguistically important. An operator A is anti-additive if it is downward monotone and satisfies the additional property that $\forall XY((AX \cap AY) \subset A(X \cup Y))$. A rule for anti-additive operators, examples of which are `no-one` and `without`, but also `not`, is easily given:

(9) If A is anti-additive:



We can continue in this vein, isolating rules connected to semantic properties that have been shown to be linguistically important. For example, [8] mentions *splittingness*, $\forall XY(A(X \cup Y) \subset (AX \cup AY))$, and *having meet*, $\forall XY((AX \cap AY) \subset A(X \cap Y))$, which we can provide with rules as follows.

(10) If A has meet:

$$\begin{array}{c} F\vec{D} : A(B \texttt{ and } C) \\ \hline F\vec{D} : AB \qquad F\vec{D} : AC \end{array}$$

(11) If A is splitting:

$$\begin{array}{c} T\vec{D} : A(B \texttt{ or } C) \\ \hline T\vec{D} : AB \qquad T\vec{D} : AC \end{array}$$

`no N` and `every N` have meet, while `some N` is splitting.

3.7 Getting Rid of Boolean Operators

Many of the rules we have seen thus far allow one to get rid of Boolean operators, even if the operator in question is not the main operator in the LLF under consideration. Here are a few more. If a Boolean is the main connective in the functor of a functor-argument expression it is of course always possible to distribute it over the argument and Booleans can likewise be pulled out of lambda-abstractions.

(12)
$$\begin{array}{ccc} X\vec{C} : (A \texttt{ and } A')B & & X\vec{C} : (\lambda x.A \texttt{ and } B) \\ | & & | \\ X\vec{C} : AB \texttt{ and } A'B & & X\vec{C} : (\lambda x.A) \texttt{ and } (\lambda x.B) \end{array}$$

These rules were given for `and`, but similar rules for `or` and `not` are also obviously correct.

Other rules that help removing Booleans from argument positions are the following.

(13) If A is mon↑:
$$\begin{array}{cc} T\vec{C}i : A(B \texttt{ and } B') & F\vec{C}i : A(B \texttt{ or } B') \\ | & | \\ T\vec{C}i : AB & F\vec{C}i : AB \\ T\vec{C}i : AB' & F\vec{C}i : AB' \end{array}$$

(14) If A is mon↓:
$$\begin{array}{cc} T\vec{C}i : A(B \texttt{ or } B') & F\vec{C}i : A(B \texttt{ and } B') \\ | & | \\ T\vec{C}i : AB & F\vec{C}i : AB \\ T\vec{C}i : AB' & F\vec{C}i : AB' \end{array}$$

It is clear that not all cases are covered, but the rules allow us to get rid of `and` and `or` at least in *some* cases.

3.8 Rules for Determiners

Let us look at rules for determiners, terms of type $\langle\langle es\rangle\langle es\rangle s\rangle$. It has often been claimed that determiners in natural language all are *conservative*, i.e. have the property $\forall XY(DXY \equiv DX(X \cap Y))$ ([2]). Leaving the question whether really *all* determiners satisfy this property aside, we can establish that for those which do we can use the following tableau rule.

(15) If D is conservative:

$$\begin{array}{c} Xi : DA(A \texttt{ and } B) \\ | \\ Xi : DAB \end{array}$$

This again is a rule that removes a Boolean operator from an argument position. Here is another. If determiners D and D' are *duals* (the pair `some` and `every` are prime examples), the following rule can be invoked. (We let $\overline{T} = F$ and $\overline{F} = T$.)

(16) If D and D' are duals:

$$\begin{array}{c} Xi : DA(\texttt{not } B) \\ | \\ \overline{X}i : D'AB \end{array}$$

The following rule applies to *contradictory* determiners, such as `some` and `no`.

(17) If D and D' are contradictories:

$$\begin{array}{c} Xi : DAB \\ | \\ \overline{X}i : D'AB \end{array}$$

There must also be rules for the logical determiners `every` and `some`. Here are some.

(18) a.

$$\begin{array}{ccc} Ti : \texttt{every } AB & & Fi : \texttt{some } AB \\ \diagup \quad \diagdown & & \diagup \quad \diagdown \\ Fci : A \quad Tci : B & & Fci : A \quad Fci : B \end{array}$$

c must be *old*.

b.

$$\begin{array}{ccc} Fi : \texttt{every } AB & & Ti : \texttt{some } AB \\ | & & | \\ Tci : A & & Tci : A \\ Fci : B & & Tci : B \end{array}$$

c must be *fresh*.

We have now entered dangerous territory, as these are *complete* rules for `some` and `every` that will certainly lead to undecidability when adopted. It may be hypothesized that the human reasoner will prefer rules such as the ones discussed before, or the obvious rule for the symmetry of `some`:

(19)

$$\begin{array}{c} Xi : \texttt{some } AB \\ | \\ Xi : \texttt{some } BA \end{array}$$

How exactly the linguistic system avoids the 'bleeding and feeding' loops that can result from the availability of rules such as those in (18) seems an important question that may partly be open to empirical investigation. Logic may provide a space of possibilities here, but only experiment can show which possibilities were nature's choice.

3.9 Further Rules

In a full paper we will add rules for the modal operators `may` and `must`, `think` and `know`. We will also consider rules that are connected to comparatives and other expressions.

4 Conclusion

One way to describe the semantics of ordinary language is by means of translation into a well-understood logical language. If the logical language comes with a model theory and a proof theory, the translation will then induce these on the fragment of language that is translated as well. A disadvantage of this procedure is that precise translation of expressions, taking heed of all their logical properties, often is difficult. Whole books have been devoted to the semantics of a few related words, but while this often was done with good reason and in some cases has led to enlightening results, describing language word by word hardly seems a good way to make progress. Tableau systems such as the one developed here provide an interesting alternative. They interface with the usual model theory, as developing a tableau can be viewed as a systematic attempt to find a model refuting the argument, but on the other hand they seem to give us a better chance in obtaining large coverage systems approximating natural logic. The format allows us to concentrate on rules that really seem linguistically important and squares well with using representations that are close to the Logical Forms in generative syntax. Tableau rules, moreover, do not only allow us to model the classical mode of reasoning, but are equally relevant for modelling alternative forms, such as abduction [1, 7] or the reasoning discussed in Johnson-Laird [14, 15], where a model of the premises is sought and the conclusion is then evaluated with respect to that model. In future work I will investigate such alternative forms of reasoning, using the natural language representations and the tableau systems presented here.

References

1. A. Aliseda-Llera. *Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence*. PhD thesis, ILLC, 1997.
2. J.F.A.K. van Benthem. Questions about Quantifiers. *Journal of Symbolic Logic*, 49:447–478, 1984.
3. J.F.A.K. van Benthem. *Essays in Logical Semantics*. Reidel, Dordrecht, 1986.
4. J.F.A.K. van Benthem. *Language in Action*. North-Holland, Amsterdam, 1991.
5. R. Bernardi. *Reasoning with Polarity in Categorial Type Logic*. PhD thesis, Utrecht University, 2002.
6. Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI, 2005.
7. M. D'Agostino, M. Finger, and D. Gabbay. Cut-Based Abduction. *Logic Journal of the IGPL*, 16(6):537–560, 2008.
8. Jaap van der Does. *Applied Quantifier Logics*. PhD thesis, University of Amsterdam, 1992.
9. D. Dowty. The Role of Negative Polarity and Concord Marking in Natural Language Reasoning. In Mandy Harvey and Lynn Santelmann, editors, *Proceedings from SALT IV*, pages 114–144. Cornell University, Ithaca, 1994.
10. J. van Eijck. Generalized Quantifiers and Traditional Logic. In J. van Benthem and A. ter Meulen, editors, *Generalized Quantifiers in Natural Language*. Foris, Dordrecht, 1985.

11. J. van Eijck. Natural Logic for Natural Language. In B. ten Cate and H. Zeevat, editors, *TbiLLC 2005*, LNAI 4363, pages 216–230. Springer-Verlag, Berlin Heidelberg, 2007.
12. F. Fyodorov, Y. Winter, and N. Francez. Order-Based Inference in Natural Logic. *Logic Journal of the IGPL*, 11(4):385–416, 2003.
13. I. Heim and A. Kratzer. *Semantics in Generative Grammar*. Blackwell, Oxford, 1998.
14. P.N. Johnson-Laird. *Mental Models: Towards a Cognitive Science of Language, Inference, and Conciousness*. Harvard University Press, Cambridge, MA, 1983.
15. P.N. Johnson-Laird. *How We Reason*. Oxford University Press, 2006.
16. G. Lakoff. Linguistics and Natural Logic. In D. Davidson and G. Harman, editors, *Semantics of Natural Language*, pages 545–665. Reidel, Dordrecht, 1972.
17. B MacCartney and C. Manning. Natural Logic for Textual Inference. In *ACL 2007 Workshop on Textual Entailment and Paraphrasing*, 2007.
18. B MacCartney and C. Manning. An Extended Model of Natural Logic. In H. Bunt, V. Petukhova, and S. Wubben, editors, *Proceedings of the 8th IWCS*, pages 140–156, Tilburg, 2009.
19. R. Montague. The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht, 1973. Reprinted in [23].
20. R.A. Muskens. *Meaning and Partiality*. CSLI, Stanford, 1995.
21. Víctor Sánchez. *Studies on Natural Logic and Categorial Grammar*. PhD thesis, University of Amsterdam, 1991.
22. F. Sommers. *The Logic of Natural Language*. The Clarendon Press, Oxford, 1982.
23. R. Thomason, editor. *Formal Philosophy, Selected Papers of Richard Montague*. Yale University Press, 1974.
24. Anna Zamansky, Nissim Francez, and Yoad Winter. A 'Natural Logic' Inference System Using the Lambek Calculus. *Journal of Logic, Language and Information*, 15:273–295, 2006.
25. F. Zwarts. Negatief-polaire Uitdrukkingen I. *Glot*, 6:35–132, 1981.