Polarity Computations in Flexible Categorial Grammar

Hai Hu
Linguistics Department
Indiana University
Bloomington, IN 47405 USA
huhai@indiana.edu

Lawrence S. Moss
Mathematics Department
Indiana University
Bloomington, IN 47405 USA
lsm@cs.indiana.edu

Abstract

This paper shows how to take parse trees in CCG and algorithmically find the polarities of all the constituents. Our work uses the well-known polarization principle corresponding to function application, and we have extended this with principles for type raising and composition. We provide an algorithm, extending the polarity marking algorithm of van Benthem. We discuss how our system works in practice, taking input from the C&C parser.

1 Introduction

The main goal of this work is to take input from text and then to automatically determine the *polarity* of all the words. For example, we aim to find the arrows in sentences like *Every dog*$^{\downarrow}$ *scares*$^{\uparrow}$ *at least two*$^{\downarrow}$ *cats*$^{\uparrow}$, *Every dog*$^{\downarrow}$ *and no cat*$^{\downarrow}$ *sleeps*$^{=}$, and *Most rabbits*$^{=}$ *hop*$^{\uparrow}$. The $^{\uparrow}$ notation means that whenever we use the given sentence truthfully, if we replace the marked word w with another word which is "$\geq w$," then the resulting sentence will still be true. So we have a *semantic inference*. The $^{\downarrow}$ notation means the same thing, except that when we substitute using a word $\leq w$, we again preserve truth. Finally, the $^{=}$ notation means that we have neither property in general; in a valid semantic inference statement, we can only replace the word with itself rather than with something larger or smaller.

For example, if we had a collection of background facts like *cats* $\leq$ *animals*, *beagles* $\leq$ *dogs*, *scares* $\leq$ *startles*, and one $\leq$ two, then our $^{\uparrow}$ and $^{\downarrow}$ notations on *Every dog*$^{\downarrow}$ *scares*$^{\uparrow}$ *at least two*$^{\downarrow}$ *cats*$^{\uparrow}$ would allow us to conclude *Every beagle startles at least one animal.*

The goal of the paper is to provide a computational system to determine the notations $\uparrow, \downarrow, =$ on input text to the best extent possible, either using hand-created parses, or output from a popular and freely available CCG parser C&C (Clark and Curran, 2007).

Using our polarity tool, we get a very easy first step on automatic inference done with little or no representation. We discuss potential applications to textual inference.

Theory We extend polarity determination for categorial grammar (CG) (see Sánchez-Valencia (1991); van Benthem (1986); van Eijck (2007); Lavalle-Martínez et al. (2017)). These papers only consider the Ajdukiewicz/Bar-Hillel (AB) flavor of CG, where the rules are restricted to application rules ($>$) and ($<$). There is a consensus that application rules alone are too restrictive to give wide-coverage grammars. We thus extend this work to the full set of *flexible combinators* used in CCG. We prove that our system is sound, in a precise sense. Further, we show how to incorporate boolean reasoning (Keenan and Faltz, 1984) to get a more complete system.

A working system We have implemented our algorithm in Python. This implementation handles sentences from the C&C parser (Clark and Curran, 2007). This is a non-trivial step on top of the theoretical advance because the parses delivered by the C&C parser deviate in several respects from the semantically-oriented input that one would like for this kind of work.

2 An Ordered Syntax-semantics Interface

The basis of the semantics is the syntax-semantics interface in formal semantics, especially in CG and CCG (Keenan and Faltz, 1984; Carpenter, 1998; Steedman, 2000; Jacobson, 2014).

Our syntax in this small paper will consist of the lexicon shown in our examples. Here is an exam-

*Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 124–129
New Orleans, June 5-6, 2018. ©2018 Association for Computational Linguistics

ple of a CCG derivation:

$$\dfrac{\dfrac{\dfrac{\textit{Fido} : n_{pr}}{\textit{Fido: } s/(s\backslash n_{pr})}\,\textsc{t} \quad \textit{ch: } (s\backslash n_{pr})/n_{pr}}{\textit{Fido chased} : s/n_{pr}}\,\textsc{b} \quad \textit{Felix} : n_{pr}}{\textit{Fido chased Felix} : s}\, > \qquad (1)$$

This tree is not the simplest one for *Fido chased Felix*. We chose it to remind the reader of the CCG rules of type-raising (T) and composition (B).

Let us fix a semantics. We first select the base types e and t. We generate complex types from these by using function types $x \to y$. We adopt a few standard abbreviations. We then fix a map from the CG categories into the types. We choose $s \mapsto t$, $n \mapsto e \to t$, $n_{pr} \mapsto e$, $np \mapsto (e \to t) \to t$, etc. (We use n_{pr} for proper names.)

A *model* $\mathcal{M}$ is a set M together with interpretations of all the lexical items by objects of the appropriate semantic type. We use M as the semantic space for the type e, $\mathbb{2} = \{\mathsf{F}, \mathsf{T}\}$ for type t, and the full set of functions for higher types. The interpretations of some words are fixed: determiners, conjunctions and relative pronouns. The model thus interprets intransitive verbs by $(et, t)t$, and transitive verbs by $(et, t)((et, t)t)$. By the Justification Theorem in Keenan and Faltz (1984), we in fact may obtain these using simpler and more natural data: for proper names we need only objects of type e, for intransitive verbs we need only et, and for transitive verbs eet.

Let S be a sentence in our fragment, and let Π be a parse tree for S. Associated to Π we have a *semantic parse tree*, giving us a term t_S in the typed lambda calculus over the base types e and t. This term may be interpreted in each model $\mathcal{M}$. For example, the interpretation corresponding to (1) is the boolean value in the model

$$(\lambda x.x[\![\textit{Fido}]\!] \circ [\![\textit{chased}]\!])[\![\textit{Felix}]\!].$$

Polarities $\uparrow$ and $\downarrow$ In order to say what the polarity symbols mean, we need to enrich our semantic spaces from sets to preorders (Moss, 2012; Icard and Moss, 2014).

A *preorder* $\mathbb{P} = (P, \leq)$ is a set P with a relation $\leq$ on P which is reflexive and transitive. Fix a model $\mathcal{M}$. Then each type x gives rise to a preorder $\mathbb{P}_x$. We order $\mathbb{P}_t$ by $\mathsf{F} < \mathsf{T}$. For $\mathbb{P}_e$ we take the flat preorder on the universe set M underlying the model. For the higher types $x \to y$, we take the set $(\mathbb{P}_x \to \mathbb{P}_y)$ of all functions and endow it with the pointwise order. In this way every one of our semantic types is naturally endowed with the structure of a preorder in every model.

A function $f : \mathbb{P} \to \mathbb{Q}$ is *monotone* (or *order preserving*) if $p \leq q$ in $\mathbb{P}$ implies $f(p) \leq f(q)$ in $\mathbb{Q}$. And f is *antitone* (or *order inverting*) if $p \leq q$ in $\mathbb{P}$ implies $f(q) \leq f(p)$ in $\mathbb{Q}$.

Each sentence S in our fragment is now interpreted in an ordered setting. This is the (mathematical) meaning of our $\uparrow$ and $\downarrow$ arrows in this paper. For example, when we write *every dog*$^{\downarrow}$ *barks*$^{\uparrow}$, this means: for all models $\mathcal{M}$, all $m_1 \leq m_2$ in $\mathbb{P}_{et}$ (for *dog*), and all $n_1 \leq n_2$ in $\mathbb{P}_{(et)t}$ (for *barks*), we have in $\mathbb{2}$ that $[\![\text{every}]\!]\ m_2\ n_1 \leq [\![\text{every}]\!]\ m_1\ n_2$.

Order-enriched types using $+$, $-$, and $\cdot$ Following Dowty (1994) we incorporate monotonicity information into the types. Function types $x \to y$ split into three versions: the monotone version $x \overset{+}{\to} y$, the antitone version $x \overset{-}{\to} y$, and the full version $x \overset{\cdot}{\to} y$. (What we wrote before as $x \to y$ is now $x \overset{\cdot}{\to} y$.) These are all preorders using the *pointwise order*. We must replace all of the ordinary slash types by versions of them which have *markings* on them.

Lexicon with order-enriched types We use S for t, N or et for $e \overset{\cdot}{\to} t = e \overset{+}{\to} t$, NP for $N \overset{\cdot}{\to} t$, NP^+ for $N \overset{+}{\to} t$, and NP^- for $N \overset{-}{\to} t$. Note that we have a different font than our syntactic types s, n, and np. Then we use $NP \overset{+}{\to} S$ for intransitive verbs, NP^+ or NP^- for noun phrases with determiners, e for proper names. For the determiners, our lexicon then uses the order-enriched types in different ways:

word	type	word	type
every	$N \overset{-}{\to} NP^+$	*no*	$N \overset{-}{\to} NP^-$
some	$N \overset{+}{\to} NP^+$	*most*	$N \overset{\cdot}{\to} NP^+$

3 Polarizing a Parse Tree

In this section, we specify the rules (see Figure 1) by which we put markings and polarities on each node of a CCG parse tree, based on a marked/order-enriched lexicon. The next section discusses the algorithm.

Input A parse tree $\mathcal{T}$ in CCG as in (1), and a marked lexicon.

Output We aim to convert $\mathcal{T}$ to a different tree $\mathcal{T}^*$ satisfying the following properties: (1) The semantic terms in $\mathcal{T}$ and $\mathcal{T}^*$ should denote the same

$$\frac{(x \xrightarrow{m} y)^{d} \quad x^{md}}{y^{d}}\,{\scriptstyle >} \qquad \frac{(x \xrightarrow{m} y)^{d} \quad (y \xrightarrow{n} z)^{md}}{(x \xrightarrow{mn} z)^{d}}\,{\scriptstyle \mathrm{B}} \qquad \frac{x^{md}}{((x \xrightarrow{m} y) \xrightarrow{+} y)^{d}}\,{\scriptstyle \mathrm{T}}$$

$$\frac{(e \to x)^{=}}{(NP \xrightarrow{+} x)^{=}}\,{\scriptstyle \mathrm{I}} \qquad \frac{(e \to x)^{d}}{(NP^{+} \xrightarrow{+} x)^{d}}\,{\scriptstyle \mathrm{J}} \qquad \frac{(e \to x)^{\mathit{flip}\, d}}{(NP^{-} \xrightarrow{+} x)^{d}}\,{\scriptstyle \mathrm{K}}$$

Figure 1: The top line contains core rules of marking and polarity. The letters m and n stand for one of the markings $+$, $-$, or $\cdot$; d stands for $\uparrow$ or $\downarrow$ (but not $=$). In (I), (J), and (K), x must be a boolean category. See charts in the text for the operations $m, d \mapsto md$ and $m, n \mapsto mn$.

function in each model. (2) The lexical items in $\mathcal{T}^*$ must receive their types from the typed lexicon. (3) The polarity of the root of $\mathcal{T}^*$ must be $\uparrow$. (4) At each node in $\mathcal{T}^*$, one of the rules in our system must be matched. Most of the rules are listed in Figure 1.

Example For $\mathcal{T}$ in (1), $\mathcal{T}^*$ could be as in (2):

$$\dfrac{\dfrac{\dfrac{\mathit{Fido}^{\uparrow} : e}{\mathit{Fido}^{\uparrow} : et \xrightarrow{+} t}\,{\scriptstyle \mathrm{T}} \qquad \mathit{chased}^{\uparrow} : e \xrightarrow{+} et}{\mathit{Fido\ chased}^{\uparrow} : e \xrightarrow{+} t}\,{\scriptstyle \mathrm{B}} \qquad \mathit{Felix}^{\uparrow} : e}{\mathit{Fido\ chased\ Felix}^{\uparrow} : t}\,{\scriptstyle >} \tag{2}$$

The signs $+$ and $-$ on the arrows are *markings*; markings apply to arrows only. We have a third marking, $\cdot$, but this does not figure into (2). Markings are used to tell if a function is interpreted (in every model) by a function which is always monotone ($+$), always antitone ($-$), or neither in general ($\cdot$). The arrows $\uparrow$ and $\downarrow$ are *polarities*. We also have a third polarity, $=$. Polarities are for specific occurrences.

Explanation of the operations on markings and polarities Each rule in Figure 1 is actually a number of other rules, and we have summarized things in terms of several operations. The chart on the left is for combining two markings m and n, and the one on the right is for combining a marking m and a polarity d, obtaining a new polarity.

$m \backslash n$	$+$	$-$	$\cdot$
$+$	$+$	$-$	$\cdot$
$-$	$-$	$+$	$\cdot$
$\cdot$	$\cdot$	$\cdot$	$\cdot$

$d \backslash m$	$+$	$-$	$\cdot$
$\uparrow$	$\uparrow$	$\downarrow$	$=$
$\downarrow$	$\downarrow$	$\uparrow$	$=$

$\mathit{flip}\uparrow = \downarrow \quad \mathit{flip}\downarrow = \uparrow$

Comments on the rules In Figure 1, x, y and z are variables ranging over marked types.

The application rule ($>$) is essentially taken from van Benthem (1986) (see also Lavalle-Martínez et al. (2017) for a survey of related algorithms); we expect that our logical system will give rise to several algorithms.

To illustrate ($>$), let us take $m = -$ and $d = \uparrow$. We then have the ($>$) rule

$$\frac{(x \xrightarrow{-} y)^{\uparrow} \quad x^{\downarrow}}{y^{\uparrow}}\,{\scriptstyle >} \tag{3}$$

This means: for all preorders $\mathbb{P}$ and $\mathbb{Q}$, all $f, g : \mathbb{P} \xrightarrow{-} \mathbb{Q}$ and all $p_1, p_2 \in P$, if $f \leq g$ and $p_2 \leq p_1$, then $f(p_1) \leq g(p_2)$.

If we were to change $x^{\downarrow}$ to $x^{\uparrow}$ in (3), we would change our statement by replacing "$p_2 \leq p_1$" with "$p_1 \leq p_2$". If we changed it to $x^{=}$, we would use "$p_1 = p_2$". In this way, we can read off a large number of true facts about preorders from our rules.

There are similar results concerning (B). Here is an example of how (B) is used, taken from (2). *Fido* has type $NP^{+} = (et) \xrightarrow{+} t$, and *chased* above it has type $NP^{+} \xrightarrow{+} (et)$. So the application of (B) results in *Fido chased* with type $NP^{+} \xrightarrow{+} t$.

The rules (I), (J), and (K) are new. In them, x must be *Boolean*. That is, it must belong to the smallest collection B containing t and with the property that if $z \in B$, then $(y \xrightarrow{\cdot} z) \in B$ for all y. B is thus the collection of types whose interpretations are naturally endowed with the structure of a *complete atomic boolean algebra* (Keenan and Faltz, 1984). Indeed, the soundness of (J) and (K) follows from the proof of the Justification Theorem (op. cit).

Figure 2 contains two applications of the (K) rules. First, the lexical entry for *chased* is $e \to et$. The first application of (K) promotes this to $NP^{-} \xrightarrow{+} et$. The NP receives a $-$ because its argument $\mathit{no\,cat}$ is of type NP^{-}. Note that the polarity flips when we do this. If we had used (J), the promotion would be to $NP^{+} \xrightarrow{+} et$, and

$$\dfrac{\dfrac{\textit{no} \quad \textit{dog}^{\downarrow}}{\textit{no dog}^{\uparrow} : NP^{-}}> \qquad \dfrac{\dfrac{\dfrac{\textit{ch}^{\uparrow} : e\to et}{\textit{ch}^{\downarrow} : NP^{-} \xrightarrow{+} et}\textsc{k} \quad \dfrac{\textit{no} \quad \textit{cat}^{\uparrow}}{\textit{no cat}^{\downarrow} : NP^{-}}>}{\textit{chased no cat}^{\downarrow} : e \to t}>}{\textit{chased no cat}^{\uparrow} : NP^{-}\xrightarrow{+} S}\textsc{k}}{\textit{no dog chased no cat}^{\uparrow} : S}<$$

Figure 2: Two applications of the (K) rules.

there would be no polarity flipping. This would be used in sentence where the object VP was *some cat* or *every cat*. The second application promoted *chased no cat* from the type *et* to $NP^{-} \xrightarrow{+} S$, again with a polarity flip. If we had used (I), we would have obtained $NP \xrightarrow{+} S$. However, this would have trivialized the polarity to $=$, and this effect would have been propagated up the tree. Rule (I) would be needed for the sentence *most dogs chased no cat*.

Several rules are not shown including "backwards" versions of (>), (B), and (T), and also versions where all polarizations are $=$. This is a technical point that is not pertinent to this short version. We should mention that due to these rules, every tree may be polarized in a trivial way, by using $=$ at all nodes. So we are really interested in the *maximally informative* polarizations, the ones that make the most predictions.

Boolean connectives, etc. We take *and* and *or* to be polymorphic of the types $B \xrightarrow{m} (B \xrightarrow{m} B)$, when B is a Boolean category and $m = +, -$, or $\cdot$. Negation flips polarities. Relative pronouns and relative clauses also can be handled. Adjectives are taken to be $N \xrightarrow{+} N$.

Other combinators This paper only discusses (T) and (B), but we also have rules for the other combinators used in CG, such as (S) and (W). For example, the (S) combinator is defined by $Sfg = \lambda x.(fx)(gx)$. In our system, the corresponding polarization rule is

$$\dfrac{(x \xrightarrow{m} (y \xrightarrow{n} z))^{d} \qquad (x \xrightarrow{mn} y)^{nd}}{(x \xrightarrow{m} z)^{d}}\ \textsc{s}$$

This combinator is part of the standard presentation of CCG, but it is less important in this paper because the C&C parser does not deliver parses using it.

4 Algorithmic Aspects

We have an algorithm[1] that takes as input a CCG tree as in (1) and outputs some tree with markings and polarities, a tree which satisfies the conditions that we have listed. The algorithm has two *phases*, similar to van Benthem's algorithm (van Benthem, 1986) for work with the Ajdukiewicz/Bar-Hillel variant of CG (only application rules). Phase 1 goes down the tree from leaves to root and adds the markings, based on the rules in Figure 1. The markings on the leaves are given in the lexicon. The rest of Phase 1 is non-deterministic. We can see this from our set of rules: there are many cases where one conclusion (on top of the line) permits several possible conclusions. As we go down the tree, we frequently need to postpone the choice.

Phase 2 of the algorithm computes the polarities, again following the rules, starting with the root. One always puts $\uparrow$ on the root, and then goes up the tree. This part of the algorithm is straightforward.

The overall algorithm is in fact non-deterministic for two reasons. As we explained, Phase 1 has a non-deterministic feature. In addition, it is always possible to polarize everything with $=$ and make similar uninformative choices for the markings. We are really interested in the *most informative* polarization, the one with the fewest number of $=$ polarities.

Soundness We have proved a *soundness theorem* for the system. Though too complicated to state in full, it might be summarized informally, as follows. Suppose we have a sentence S in English, and suppose that the lexical items in S are given semantics that conform to our assumptions. (This means that the semantics of the lexical entries must belong to the appropriate types.) Then any semantic statement about the $^{\uparrow}$, $^{\downarrow}$, $^{=}$ marking predicted by our system is correct. See Moss (2018) for details.

Completeness We have not proven the completeness of our system/algorithm, and indeed this is an open question. What completeness would mean for a system like ours is that whenever we have an input CCG parse tree and a polarization of its words which is semantically valid in the sense that it holds no matter how the nouns, verbs, etc. are interpreted, then our algorithm would detect this. This completeness would be a property

[1] https://github.com/huhailinguist/ccg2mono

of the rules and also of the polarization algorithm. The experience with similar matters in Icard and Moss (2013) suggests that completeness will be difficult.

Efficiency of our algorithm Our polarization is quite fast on the sentences which we have tried it on. We conjecture that it is in polynomial time, but the most obvious complexity upper bound to the polarization problem is NP. The reason that the complexity is not "obviously polynomial" is that for each of the type raising steps in the input tree, one has three choices of the raise. In more detail, suppose that the input tree contains

$$\frac{x}{(x \to y) \to y}\ \text{T}$$

Then our three choices for marking are: $(x \overset{+}{\to} y) \overset{+}{\to} y$, $(x \overset{-}{\to} y) \overset{+}{\to} y$, and $(x \overset{\cdot}{\to} y) \overset{+}{\to} y$. Our implementation defers the choice until more of the tree is marked. But prima facie, there are an exponential number of choices. All of these remarks also apply to the applications of (I), (J), and (K); these do not occur in the input tree, and the algorithm must make a choice somehow. Thus we do not know the worst-case complexity of our algorithm.

5 What Our System Can Currently Do

We tokenized input sentences using the script from the ccg2lambda system (Martínez-Gómez et al., 2016). The tokenized sentences were then parsed using the C&C parser (Clark and Curran, 2007), which is trained on the CCGbank (Hockenmaier and Steedman, 2007). Then we run our algorithm.

We are able to take simple sentences all the way through. For example, our system correctly determines the polarities in

No$^{\uparrow}$ *man*$^{\downarrow}$ *walks*$^{\downarrow}$
Every$^{\uparrow}$ *man*$^{\downarrow}$ *and*$^{\uparrow}$ *some*$^{\uparrow}$ *woman*$^{\uparrow}$ *sleeps*$^{\uparrow}$
Every$^{\uparrow}$ *man*$^{\downarrow}$ *and*$^{\uparrow}$ *no*$^{\uparrow}$ *woman*$^{\downarrow}$ *sleeps*$^{=}$
If$^{\uparrow}$ *some*$^{\downarrow}$ *man*$^{\downarrow}$ *walks*$^{\downarrow}$, *then*$^{\uparrow}$ *no*$^{\uparrow}$ *woman*$^{\downarrow}$ *runs*$^{\downarrow}$
Every$^{\uparrow}$ *man*$^{\downarrow}$ *does*$^{\downarrow}$ *n't*$^{\uparrow}$ *hit*$^{\downarrow}$ *every*$^{\downarrow}$ *dog*$^{\uparrow}$
No$^{\uparrow}$ *man*$^{\downarrow}$ *that*$^{\downarrow}$ *likes*$^{\downarrow}$ *every*$^{\downarrow}$ *dog*$^{\uparrow}$ *sleeps*$^{\downarrow}$
Most$^{\uparrow}$ *men*$^{=}$ *that*$^{=}$ *every*$^{=}$ *woman*$^{=}$ *hits*$^{=}$ *cried*$^{\uparrow}$
Every$^{\uparrow}$ *young*$^{\downarrow}$ *man*$^{\downarrow}$ *that*$^{\uparrow}$ *no*$^{\uparrow}$ *young*$^{\downarrow}$ *woman*$^{\downarrow}$
    hits$^{\uparrow}$ *cried*$^{\uparrow}$

As shown, our algorithm polarizes all words in the input. For determiners, this actually is useful. It is (arguably) background knowledge, for example that *every* $\leq$ *some*; *at least two* $\leq$ *at least one* $\equiv$ *some*, *no* $\leq$ *at most one* $\leq$ *at most two*, etc. These would not be part of the algorithm in this paper, but rather they would be background facts that figure into inference engines built on this work.

Problems Our end-to-end system is sound in the sense that it polarizes the correctly input semantic representations. However, it is limited by the quality of the parses coming from the C&C parser. While the parser has advantages, its output is sometimes not the optimal for our purposes. For example, it will assign the supertag `N/N` to *most*, but `NP/N` to other quantifiers. Thus in order to handle *most*, one has to manually change the parse trees. It also parses relative clauses as *(no dog) (who chased a cat) died* rather than *(no (dog who chased a cat)) died*. Furthermore, the parser sometimes behaves differently on intransitive verbs likes *walks* than on *cries*. Currently, we manually fix the trees when they systematically deviate from our desired parses (e.g. relative clauses). Finally, as with any syntactic parser, it only delivers one parse. So ambiguous sentences are not treated in any way by our work.

6 Future Work: Inference, and Connections with Other Approaches

We certainly plan to use the algorithm in connection with *inference*, since this has always been the a primary reason to study monotonicity and polarity. Indeed, once one has correct polarity markings, it is straightforward to use those to do inference from any background facts which can be expressed as inequalities. This would cover taxonomic statements like *dog* $\leq$ *animal* and also predications like *John isa swimmer*. Our future work will show logical systems built this way.

Connections This paper invites connections to other work in the area, especially MacCartney and Manning (2009) and Nairn et al. (2006), which shared similar aims as ours, but were not done in the CCG context. We also think of work on automatic discovery of downward-entailments (Cheung and Penn, 2012; Danescu et al., 2009), and other work on natural logic (Fyodorov et al., 2003; Zamansky et al., 2006; Moss, 2015; Abzianidze, 2017). Additionally, our work could be incorporated in several ways into textual entailment systems (e.g. Dagan et al., 2013).

References

Lasha Abzianidze. 2017. Langpro: Natural language theorem prover. *CoRR*, abs/1708.09417.

Johan van Benthem. 1986. *Essays in Logical Semantics*. Reidel, Dordrecht.

Bob Carpenter. 1998. *Type-Logical Semantics*. MIT Press.

Jackie Cheung and Gerald Penn. 2012. Unsupervised detection of downward-entailing operators by maximizing classification certainty. In *Proc. 13th EACL*, pages 696–705.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Cristian Danescu, Lillian Lee, and Richard Ducott. 2009. Without a 'doubt'? Unsupervised discovery of downward-entailing operators. In *Proceedings of NAACL HLT*.

David Dowty. 1994. The role of negative polarity and concord marking in natural language reasoning. In *Proceedings of Semantics and Linguistic Theory (SALT) IV*.

Jan van Eijck. 2007. Natural logic for natural language. In *Logic, Language, and Computation*, volume 4363 of *LNAI*, pages 216–230. Springer-Verlag.

Yaroslav Fyodorov, Yoad Winter, and Nissim Fyodorov. 2003. Order-based inference in natural logic. *Log. J. IGPL*, 11(4):385–417. Inference in computational semantics: the Dagstuhl Workshop 2000.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Thomas F. Icard and Lawrence S. Moss. 2013. A complete calculus of monotone and antitone higher-order functions. In *Proceedings, TACL 2013*, volume 23 of *EPiC Series*, pages 96–99. Vanderbilt University.

Thomas F. Icard and Lawrence S. Moss. 2014. Recent progress on monotonicity. *Linguistic Issues in Language Technology*, 9(7):167–194.

Pauline Jacobson. 2014. *An Introduction to the Syntax/Semantics Interface*. Oxford University Press.

Edward L. Keenan and Leonard M. Faltz. 1984. *Boolean Semantics for Natural Language*. Springer.

José-de-Jesús Lavalle-Martínez, Manuel Montes-y Gómez, Luis Villaseñor-Pineda, Héctor Jiménez-Salazar, and Ismael-Everardo Bárcenas-Patiño. 2017. Equivalences among polarity algorithms. *Studia Logica*.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *IWCS-8, Proceedings of the Eighth International Conference on Computational Semantics*, pages 140–156.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL 2016 System Demonstrations*, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Lawrence S. Moss. 2012. The soundness of internalized polarity marking. *Studia Logica*, 100(4):683–704.

Lawrence S. Moss. 2015. Natural logic. In *Handbook of Contemporary Semantic Theory, Second Edition*, pages 646–681. Wiley-Blackwell.

Lawrence S. Moss. 2018. Foundations of polarity determination for flexible categorial grammars. Unpublished ms.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing relative polarity for textual inference. In *Proceedings of ICoS-5 (Inference in Computational Semantics)*, Buxton, UK.

Victor Sánchez-Valencia. 1991. *Studies on Natural Logic and Categorial Grammar*. Ph.D. thesis, Universiteit van Amsterdam.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Anna Zamansky, Nissim Francez, and Yoad Winter. 2006. A 'natural logic' inference system using the Lambek calculus. *Journal of Logic, Language, and Information*, 15(3):273–295.