Recent Progress on Monotonicity

THOMAS F. ICARD, III[1] AND LAWRENCE S. MOSS[2]

This paper serves two purposes. It is a summary of much work concerning formal treatments of monotonicity and polarity in natural language, and it also discusses connections to related work on exclusion relations, and connections to psycholinguistics and computational linguistics. The second part of the paper presents a summary of some new work on a formal Monotonicity Calculus.

1 Introduction

It has been known since Aristotle that many entailment patterns between linguistic expressions can be derived by simple predicate replacement. *Monotonicity reasoning*, involving a particular sort of predicate replacement, has played an especially important role in the development of logic and semantics, and continues to do so today.

Monotonicity is a pervasive feature of natural language, and it has been linked to many fundamental aspects of linguistic processing, reasoning, and even grammar. In the late 1980s van Benthem (1986, 1991) and Sánchez-Valencia (1991) defined proof systems for reasoning about entailment using monotonicity in higher-order languages. Working in a simply-typed language and building on the long line of work in categorial grammars, the idea behind the so-called Monotonicity Calculus was to mark expressions of functional type with monotonicity information, and use this in a proof system. Work on this topic has recently been revived by computational linguists, and it has found its way into

[1]Department of Philosophy, Stanford University.
[2]Department of Mathematics, Indiana University, Bloomington. This work was partially supported by a grant from the Simons Foundation (#245591 to Lawrence Moss).

LiLT Volume 9
Perspectives on Semantic Representations for Textual Inference.

important applications (see Section 4).

This paper offers an overview of recent developments in the study of monotonicity in natural language. Strictly speaking, monotonicity (and its opposite *antonicity*) is a property of functions, and is therefore a semantic notion. However, we will also be concerned with what is usually called *polarity*, a syntactic notion. In fact, the Monotonicity Calculus and related logical systems can be thought of as exploiting the close connection between polarity and monotonicity. We use syntactic markings to define proof systems whose soundness is guaranteed by the correspondences between positive polarity and monotonicity on the one hand, and negative polarity and antitonicity on the other. One of the main contributions of this paper is to make these notions fully precise, and to present a completeness result for the Monotonicity Calculus.

The outline of the paper is as follows. Section 2 is a less technical introduction to the fundamental concepts involved. In Sections 2.1-2.3, we introduce the notions of *monotonicity*, *antitonicity*, and *polarity*. Using examples from natural language semantics and basic algebra, we illustrate the use of marked types, and explain how expressions of these types are interpreted model-theoretically. We also outline the fundamental algorithms used for marking parsed expressions with polarity information. In Section 2.5, we discuss a variation on this approach, called *internalized polarity marking*. Section 3 contains a short summary of recent work on extending monotonicity calculi to reasoning about exclusion relations in addition to inclusion relations.

In Section 4 we offer a brief summary of recent work in psychology and computational linguistics that has made critical use of earlier theoretical work on monotonicity. Finally Section 5 presents formal details underpinning the Monotonicity Calculus, including explicit development of a suitable type system (Section 5.1), a precise statement of the proof rules (Section 5.4), and a discussion of soundness and completeness (Section 5.5). We omit some technical details and proofs, which appear in a companion manuscript in progress (Icard and Moss 2013).

2 Reasoning about Monotonicity

2.1 Basic Definitions and Examples

Monotonicity is a pervasive feature in natural language inference, and it is not only interesting in its own right but is tied up with other features of language. However, it is not always so easy to get a handle on what monotonicity actually *is*, to say what it comes to semantically. Perhaps the clearest explanation takes monotonicity to be a property of functions defined on pre-orders. A *pre-order* is a pair $(D, \leq)$ consisting of a set D and a relation $\leq$ on D which is reflexive ($d \leq d$ for all $d \in D$) and transitive (if $c \leq d$ and $d \leq e$, then also $c \leq e$). If we have two (pre-)ordered domains $(D_1, \leq_1)$ and $(D_2, \leq_2)$, a function $f : D_1 \to D_2$ is *monotonic* if, whenever $a \leq_1 b$ we have $f(a) \leq_2 f(b)$. We say f is *antitonic* if, whenever $a \leq_1 b$ it follows that $f(b) \leq_2 f(a)$.

This paper is concerned with two general examples: the first, pertaining to language, is close to what we see in areas of formal semantics, and is related to work in Montague grammar and the theory of generalized quantifiers. The second is a mathematical example that in some ways is a variation on the first.

Example 1. Consider a word like every. If we think of every as taking two arguments, then intuitively it is a function antitonic in its first argument and monotonic in its second argument. For a sentence such as, Every aardvark sees a hyena, if we replace aardvark with the more specific brave aardvark, the resulting sentence is entailed by the first: Every brave aardvark sees a hyena. Likewise, if we replace hyena by the less specific carnivore, then this sentence is also entailed: Every aardvark sees a carnivore. Both of these facts can easily be seen to follow from the standard interpretation of every as the subset relation on predicates.

Indeed, each quantifier in English has its own *monotonicity profile*, where $+$ means monotonic, $-$ antitonic, and $\cdot$ neither monotonic nor antitonic in general:

$-$ every $+$	$+$ not every $-$	$\cdot$ exactly n $\cdot$
$+$ some $+$	$\cdot$ most $+$	$+$ at least n $+$
$-$ no $-$	$\cdot$ few $-$	$-$ at most n $-$

Here is how to read this notation, starting with the first example of every. The idea is that in a sentence of the form every A B, if we replace A by $C \subseteq A$, then every A B entails every C B (Example 1). Similarly if we replace B by $D \supseteq B$, then every A B entails every A D. The $-$ indicates that the sentence as a whole is antitonic in A; similarly, the $+$ indicates that the sentence is monotonic in B. The monotonicity profiles above are the strongest possible statements. That is, we could have put $\cdot$ in all the profiles. But then this would mean that we failed to recognize important information. To put things differently, we can read $\cdot$ as saying neither monotonic nor antitonic in general.

Expressions other than determiners also have interesting monotonicity profiles. For instance, a preposition like 'without' is clearly antitonic: without a doubt entails without a reasonable doubt. Moreover, we can reason about expressions with embedded operators:

(1)

	No aardvark without a keen sense of smell can find food
implies	No aardvark without a sense of smell can find food

Here sense of smell is embedded under three operators: a, without, and no. Since keen sense of smell "implies" sense of smell, it follows that without a sense of smell "implies" without a keen sense of smell. Applying no reverses this once more.

So far we are treating the "implies" relation between English expressions informally. Between predicates this is assumed to be the "more specific than" relation; between sentences it is the "entails" relation. This can be made more precise by assigning semantic *types* to English expressions and interpreting typed expressions in appropriately ordered domains. A precise type system will be defined later in Section 5.1. The following is a mathematical example, illustrating domains for interpreting a simple mathematical language.

Example 2. Let $D_1 = D_2 = \mathbb{R}$ be the real numbers, and $\leq_1 = \leq_2 = \leq$ the ordinary "less than or equal to" relation on $\mathbb{R}$. Then it is easy to check that, e.g., as functions of a variable x, the functions 2^x and $7+x$ are monotonic, while $-x$ is antitonic. As an example of monotonicty reasoning, consider how one might determine which of the following two expressions is larger: $-(7+2^{-3})$ or $-(7+2^{-4})$? One way to do it would simply be to evaluate both sides and then compare. This is *not* of interest to us, because it avoids the general principles that are our main topic. Instead, we can argue as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{3<4}{-4<-3}\;-x\text{ is antitone}}{2^{-4}<2^{-3}}\;2^x\text{ is monotone}}{7+2^{-4}<7+2^{-3}}\;7+x\text{ is monotone}}{-(7+2^{-3})<-(7+2^{-4})}\;-x\text{ is antitone}$$

Naturally, linguistic examples are more relevant for our purposes than mathematical examples. However, it will often simplify the presentation to see the mathematical examples.

2.2 Monotonicity in algebra *via* grammatical inference

Consider the function

$$f(v,w,x,y,z) \quad = \quad \frac{x-y}{2^{z-(v+w)}}.$$

We intend this to be a function from five real numbers back to the reals. Suppose we fix numerical values for the variables v, w, x, y, z, and then we take each variable in turn, and (while keeping the others fixed) increase its value. For v, w, and x, the overall value goes up. For y and z, it goes down. We would summarize all of the observations by:

$$f(v^+, w^+, x^+, y^-, z^-).$$

The notations $^+$ and $^-$ are what we mean by *polarities.* Note that these really are properties of *occurrences* of variables: a given variable might have both positive and negative occurrences in a given function. For example, consider x in $(x+1)/(x+2)$.

Here is how we can think about this in terms close to the way formal semantics works with the simply typed lambda calculus. We take a single type r, and then we have function symbols

(2)
$$\begin{array}{ll} \mathsf{plus} : r \to (r \to r) & \mathsf{minus} : r \to (r \to r) \\ \mathsf{times} : r \to (r \to r) & \mathsf{div2} : r \to (r \to r) \end{array}$$

The variables v, w, …, z may be taken as constants of type r.

We should mention that the natural semantics of these symbols are going to be *higher-order one-place* functions, rather than the more usual binary functions. For example, $[\![\mathsf{plus}]\!]$ is the function from $\mathbb{R}$ to functions from $\mathbb{R}$ to itself which takes a real number a to the function $\lambda b.a+b$. So we would write $(\mathsf{plus}(x))(y)$ to indicate $x+y$. Dropping the parentheses and writing $\mathsf{plus}\ x\ y$ allows one to read through the higher-order functions, but officially they are still going to be there. See Example 10 for more on the semantics of our syntax for algebraic expressions. We also must mention that $\mathsf{div2}$ is *not* supposed to be one-place version of the usual division operation. The idea is that $\mathsf{div2}(x)(y)$ should be $x \div 2^y$, not x/y. This complication is to make everything monotone. We obtain terms in Polish notation:

(3)
$$\dfrac{\dfrac{\mathsf{div2} : r \to (r \to r) \qquad \dfrac{\dfrac{\mathsf{minus} : r \to (r \to r) \quad x : r}{\mathsf{minus}\ x : r \to r} \qquad y : r}{\mathsf{minus}\ x\ y : r}}{\mathsf{div2}\ \mathsf{minus}\ x\ y : r \to r} \qquad \dfrac{\dfrac{\mathsf{minus} : r \to (r \to r) \quad z : r}{\mathsf{minus}\ z : r \to r} \qquad t}{\mathsf{minus}\ z\ \mathsf{plus}\ v\ w : r}}{\mathsf{div2}\ \mathsf{minus}\ x\ y\ \mathsf{minus}\ z\ \mathsf{plus}\ v\ w : r}$$

where t above is $\mathsf{minus}\ z\ \mathsf{plus}\ v\ w$ with the derivation

(4)
$$\dfrac{\dfrac{\mathsf{minus} : r \to (r \to r) \quad z : r}{\mathsf{minus}\ z : r \to r} \qquad \dfrac{\dfrac{\mathsf{plus} : r \to (r \to r) \quad v : r}{\mathsf{plus}\ v : r \to r} \qquad w : r}{\mathsf{plus}\ v\ w : r}}{\mathsf{minus}\ z\ \mathsf{plus}\ v\ w : r}$$

The term in (3) corresponds to the term at the beginning of this section, $(x-y)/2^{z-(v+w)}$, and the term in (4) corresponds to the subterm $z-(v+w)$. We are presenting a derivation of the term in the style familiar from categorial grammar (CG), and we assume that the reader has some familiarity with these ideas. In fact, we will only appeal to the *elimination* rules (the application of functions to arguments) in this paper, which will mostly be suppressed.

We are interested in a term corresponding to f from above. For reasons of space, we leave the type information implicit:

$$\dfrac{\dfrac{\mathsf{div2} \qquad \dfrac{\dfrac{\mathsf{minus}\quad x}{\mathsf{minus}\ x}\quad y}{\mathsf{minus}\ x\ y}}{\mathsf{div2\ minus}\ x\ y} \qquad \dfrac{\dfrac{\mathsf{minus}\quad z}{\mathsf{minus}\ z} \qquad \dfrac{\dfrac{\mathsf{plus}\quad v}{\mathsf{plus}\ v}\quad w}{\mathsf{plus}\ v\ w}}{\mathsf{minus}\ z\ \mathsf{plus}\ v\ w}}{\mathsf{div2\ minus}\ x\ y\ \mathsf{minus}\ z\ \mathsf{plus}\ v\ w} \tag{5}$$

Polarity determination The central question is whether we can determine the polarities of the variables from the tree representation. Here is one presentation of the algorithm proposed by van Benthem (1986).

1. Label the root with $+$.
2. Propagate notations up the tree. The right branches of nodes for $\mathsf{div2}$ and minus of type r flip notations. Otherwise, we maintain the notations as we go up the tree.

For example, here is how this works with the term in (5):

$$\dfrac{\dfrac{\mathsf{div2}^{+} \qquad \dfrac{\dfrac{\mathsf{minus}^{+}\quad x^{+}}{(\mathsf{minus}\ x)^{+}}\quad y^{-}}{(\mathsf{minus}\ x\ y)^{+}}}{(\mathsf{div2\ minus}\ x\ y)^{+}} \qquad \dfrac{\dfrac{\mathsf{minus}^{-}\quad z^{-}}{(\mathsf{minus}\ z)^{-}} \qquad \dfrac{\dfrac{\mathsf{plus}^{+}\quad v^{+}}{(\mathsf{plus} : v)^{+}}\quad w^{+}}{(\mathsf{plus} : v\ w)^{+}}}{(\mathsf{minus}\ z\ \mathsf{plus}\ v\ w)^{-}}}{(\mathsf{div2\ minus}\ x\ y\ \mathsf{minus}\ z\ \mathsf{plus}\ v\ w)^{+}} \tag{6}$$

Notice that the polarity markings on the variables agree with what we saw before: $f(v^{+}, w^{+}, x^{+}, y^{-}, z^{-})$.

The algorithm was first proposed in categorial grammar in van Benthem (1986) to formalize the $+$ and $-$ notation (he used $\uparrow$ for $+$ and $\downarrow$ for $-$). His proposal was then worked out by Sánchez-Valencia (1991).

2.3 New types

We have seen the function symbols plus, ..., $\mathsf{div2}$ in (2) along with their types as higher order functions. In the polarity determination algorithm described above, we started with the type declarations in (2) and then used the extra monotonicity features of the lexical items.

However, we could also proceed in a different manner. We could record additional monotonicity/antitonicity information concerning our symbols *into the types*. For example, minus is monotone in its first argument and antitone in its second. We therefore elaborate on (2) with a different set of type declarations:

$$(7)\qquad \begin{array}{ll} \mathsf{plus} : r \overset{+}{\to} (r \overset{+}{\to} r) & \mathsf{minus} : r \overset{+}{\to} (r \overset{-}{\to} r) \\ \mathsf{times} : r \overset{+}{\to} (r \overset{+}{\to} r) & \mathsf{div2} : r \overset{+}{\to} (r \overset{-}{\to} r) \end{array}$$

We can be more general by using the types in (7) directly. We are therefore reconsidering the *syntax* of the algebraic expressions. For example, we revisit (4) in this new regime:

$$\dfrac{\dfrac{\mathsf{minus} : r \overset{+}{\to} (r \overset{-}{\to} r) \quad z : r}{\mathsf{minus}\ z : r \overset{-}{\to} r} \qquad \dfrac{\dfrac{\mathsf{plus} : r \overset{+}{\to} (r \overset{+}{\to} r) \quad v : r}{\mathsf{plus}\ v : r \overset{+}{\to} r} \qquad w : r}{\mathsf{plus}\ v\ w : r}}{\mathsf{minus}\ z\ \mathsf{plus}\ v\ w : r}$$

We wish to understand the connection of the new syntactic types to something in the semantics, and so we introduce new *type domains.* These will be preorders, not simply unstructured sets.

$$\begin{array}{lcl} \mathbb{D}_r & = & \mathbb{R}, \text{ the real numbers with the usual order } \leq \\ \mathbb{D}_{r \overset{+}{\to} r} & = & \text{the monotone functions from } \mathbb{D}_r \text{ to } \mathbb{D}_r \\ \mathbb{D}_{r \overset{-}{\to} r} & = & \text{the antitone functions from } \mathbb{D}_r \text{ to } \mathbb{D}_r \\ \mathbb{D}_{r \overset{+}{\to} (r \overset{+}{\to} r)} & = & \text{the monotone functions from } \mathbb{D}_r \text{ to } \mathbb{D}_{r \overset{+}{\to} r} \\ \mathbb{D}_{r \overset{+}{\to} (r \overset{-}{\to} r)} & = & \text{the monotone functions from } \mathbb{D}_r \text{ to } \mathbb{D}_{r \overset{-}{\to} r} \end{array}$$

The natural interpretations of the functions plus, minus, etc., belong to the appropriate domains. However, we will not be concerned with the precise functions these terms denote, only with whether they are monotone or antitone. Notice all of our functions in this example are either monotone or antitione.

The issue once again is how to determine the polarities of the individual occurrences. We re-state the bottom-up (root-to-leaves) algorithm that we saw before, this time in a more general form. (In our statement, the root is at the "bottom" of the tree, and the parents are "above" their child.)

1. Label the root with $+$.
2. Propagate notations up the tree.
   (a) If a node is labeled ℓ and its parents are of type $\sigma \overset{+}{\to} \tau$ and σ, then both parents are labeled ℓ.
   (b) If a node is labeled ℓ and its parents are of type $\sigma \overset{-}{\to} \tau$ and σ, then the former parent is to be labeled ℓ and the latter parent is to be labeled $-\ell$, that is, the flipped version of ℓ.

The point is to present the algorithm using only the $+$ and $-$ signs on the arrows, rather than the particular lexical items used. That is, the

$+$ and $-$ signs on the type arrows encode all the information that the polarity algorithm would use. We simply ignore anything else we might know about these functions. All inference is driven by monotonicty information alone.

A variation on the polarity determination presented here was proposed by van Eijck (2007), who observed that the same result can be achieved by marking nodes in the syntax tree based on whether they *respect* or *flip* the markings.

2.4 Operations which are neither monotone nor antitone

Not every operation of interest is monotone or antitone. We have already seen that the semantics of most would be neither monotone nor antitone in its first argument. For an easy mathematical example, consider the absolute value function $|x| : \mathbb{R} \to \mathbb{R}$. To continue our treatment, we take a function symbol abs : $r \overset{\cdot}{\to} r$. We write $\overset{\cdot}{\to}$ because, while the interpretation is a function, we cannot classify this interpretation as monotone or antitone. More formally, we can say that the interpretation of abs belongs to the set $\mathbb{D}_r$ given as follows:

$$\mathbb{D}_{r \overset{\cdot}{\to} r} \quad = \quad \text{the set of all functions from } \mathbb{D}_r \text{ to } \mathbb{D}_r$$

That is, all we know about abs is that it is some function from $\mathbb{R}$ to $\mathbb{R}$.

We can extend the algorithm above in a straightforward way. Perhaps the most elegant extension involves considering the set $\mathcal{M} = \{+, -, \cdot\}$ of *markings* on the arrows to be an algebraic structure, using the operation $\circ$ defined in the table below:

$\circ$	$+$	$-$	$\cdot$
$+$	$+$	$-$	$\cdot$
$-$	$-$	$+$	$\cdot$
$\cdot$	$\cdot$	$\cdot$	$\cdot$

This tiny algebra is the basis of much work on natural logic (van Benthem 2008; Sánchez-Valencia 1991; van Eijck 2007; Zamansky et al. 2006). We can generalize the bottom-up algorithm in point 2(b):

2(b) If a node is labeled ℓ and its parents are of type $\sigma \overset{m}{\to} \tau$ and σ, then the former parent is to be labeled ℓ and the latter parent is to be labeled $m \circ \ell$.

2.5 Internalized types

At this point, we have seen several ways to determine polarities in a parse tree given by a categorial grammar. Either way, the determination of polarities is an *external* feature of the syntax tree, something determined by an algorithm. Instead of complicating the architecture of grammar, we could complicate the particular grammar that we use and achieve the same thing. We introduce *negative signs on types*, to denote *opposite preorders*. And we allow a lexical item to have more than one type. (This is standard in categorial grammar.) We use the following lexicon:

$$\begin{array}{ll}
v, w, x, y, z : r & v, w, x, y, z : -r \\
\mathsf{plus} : r \to (r \to r) & \mathsf{plus} : -r \to (-r \to -r) \\
\mathsf{minus} : r \to (-r \to r) & \mathsf{minus} : -r \to (r \to -r) \\
\mathsf{times} : r \to (r \to r) & \mathsf{times} : -r \to (-r \to -r) \\
\mathsf{div2} : r \to (-r \to r) & \mathsf{div2} : -r \to (r \to -r)
\end{array}$$

Here is an explanation of what the types refer to, along the lines of what we did above.

$\mathbb{D}_r$	$=$	$\mathbb{R}$, the real numbers with the usual order $\leq$
$\mathbb{D}_{-r}$	$=$	$\mathbb{R}$, the real numbers with the opposite order $\geq$
$\mathbb{D}_{r\to r}$	$=$	the monotone functions from $\mathbb{D}_r$ to $\mathbb{D}_r$
$\mathbb{D}_{r\to -r}$	$=$	the monotone functions from $\mathbb{D}_r$ to $\mathbb{D}_{-r}$
	$=$	the antitone functions from $\mathbb{D}_r$ to $\mathbb{D}_r$
$\mathbb{D}_{-r\to(r\to -r)}$	$=$	the monotone functions from $\mathbb{D}_{-r}$ to $\mathbb{D}_{r\to -r}$

A term corresponding to $z/2^{x-y}$ parses as

$$\dfrac{\dfrac{\mathsf{div2} : r \to (-r \to r) \quad z : r}{\mathsf{div2}\ z : -r \to r} \qquad \dfrac{\dfrac{\mathsf{minus} : -r \to (r \to -r) \quad x : -r}{\mathsf{minus}\ x : r \to -r} \quad y : r}{\mathsf{minus}\ x\ y : -r}}{\mathsf{div2}\ z\ \mathsf{minus}\ x\ y : r}$$

The parse tree automatically indicates the polarities. For example, the term as a whole is antitone in x, since its type is $-r$; there is no parse of the term which has $x : r$ as a leaf, even though this typing is available in the grammar. Similarly, the term is monotone in y and z. The point here is that there is no need to have a separate algorithm for polarity determination.

For linguistic reasons for this *internalized* line of work, see Dowty (1994). For more on this flavor of monotonicity in categorial grammar, see Moss (2012).

In the rest of this paper, we are not going to discuss the internalized approach. One reason for this is that we are interested in presenting an account that also allows functions to be labeled as neither monotone nor antitone, as explained above in Section 2.4.

2.6 Monotonicity reasoning

Up until now, we have mainly been concerned with developing tools which allow one to look at a syntactic representation and see which positions are monotone, which antitone, and which neither. But this is only the beginning. As we mentioned at the outset of this paper, much of the interest in monotonicity is connected with *inference*.

Here is an example of what we are after in this regard; the formal details will appear in Section 5 below. Suppose that we have acquired some lexical monotonicity information, for example that

$$\begin{array}{c}\mathsf{run} \leq \mathsf{move}\\ \mathsf{cat} \leq \mathsf{animal}\end{array}$$

We also assume we have the monotonicity information that we codify in the typing for the determiner every and for a transitive verb like see. Independent of how an agent would learn any of this information, we explain how these facts could be exploited in simple reasoning. For instance, how is it that

(8) Everything which sees every cat runs
implies Everything which sees every animal moves

Indeed, we would like the inference to be formalized in the same kind of proof-theoretic manner that we saw with derivation trees in categorial grammar in the first place; these are derivations in *natural deduction*. The difference is that in addition to inference between sentences, we want a general notion of $\leq$ between items of any syntactic category. What we have written here is on the level of noun phrases, and on the VP level it would be

$$\mathsf{see\ every\ animal} \leq \mathsf{see\ every\ cat}$$

This is the first step in our semi-formal derivation below, going from the assumption (the leaf of the tree) to the line below it.

$$\dfrac{\dfrac{\mathsf{cat} \leq \mathsf{animal}}{\mathsf{see\ every\ animal} \leq \mathsf{see\ every\ cat}}}{\mathsf{Everything\ which\ sees\ every\ cat\ runs} \leq \mathsf{Everything\ which\ sees\ every\ animal\ runs}}$$

Notice that both steps are a kind of antitonicity: the positions of animal and cat have switched. Going further, we might like to combine the derivation above with one involving runs and moves to give an account of the inference in (8). By the monotonicity of the second argument of every, we have:

$$\dfrac{\mathsf{runs} \leq \mathsf{moves}}{\mathsf{Everything\ which\ sees\ ev.\ animal\ runs} \leq \mathsf{Everything\ which\ sees\ ev.\ animal\ moves}}$$

Finally, if we let

$t =$ Everything which sees every cat runs

$u =$ Everything which sees every animal runs

$v =$ Everything which sees every animal moves

then from $t \leq u$ (first derivation) and $u \leq v$ (second derivation), we derive $t \leq v$, i.e., the entailment in (8).

All of these steps can be formalized in a Monotonicity Calculus that we shall see in Section 5. Here is a preview of the rules:

$$(\textsc{Refl})\ \frac{}{t \leq t} \qquad (\textsc{Trans})\ \frac{t \leq u \qquad u \leq v}{t \leq v}$$

$$(\textsc{Mono})\ \frac{u \leq v}{t[u^+] \leq t^{v \leftarrow u}} \qquad (\textsc{Anti})\ \frac{u \leq v}{t^{v \leftarrow u} \leq t[u^-]}$$

$$(\textsc{Point})\ \frac{s \leq t}{s(u) \leq t(u)}$$

See Section 5 for details. The names are for the evident abbreviations: Refl for *reflexive*, Trans for *transitive*, Mono for *monotone*, Anti for *antitone*, and Point for *pointwise*.

Finally, note that in natural language this kind of reasoning is not limited to *predicate* restriction and expansion. It also includes reasoning about events, locations, times, sums, and more general mereological domains. All of these have natural associated pre-orders (often with more structure), and we can consider monotone and antitone functions over such domains. For instance, lives in is monotonic with respect to the "part-of" relation on locations, whereas lasts more than is antitonic with respect to the "subinterval" relation on times. Moreover, these interact with the quantifiers in the expected ways. For instance:

$$\cfrac{\cfrac{\cfrac{\text{2 hours} \leq \text{6 hours}}{\text{more than 6 hours} \leq \text{more than 2 hours}}}{\text{play that lasts more than 6 hours} \leq \text{play that lasts more than 2 hours}}}{\text{Every play...more than 2 hrs is too long} \leq \text{Every play...more than 6 hrs is too long}}$$

For simplicity, our natural language examples will be based on the simple model of individuals, predicates and properties (sets of individuals), quantifiers, and so on. But it is important to point out that the sort of reasoning we are describing is quite general.

3 Reasoning about Exclusion

Monotonicity reasoning certainly does not exhaust the inferential patterns that follow from the standard logical interpretations of natural language expressions. An obvious question is whether we can go further with the "surface reasoning" approach of marking types with useful inferential information. Recent work by MacCartney (2009) and

MacCartney and Manning (2009) has shown that many new inferential patterns can be derived by tracking how functional expressions project *exclusion* relations, in addition to the *inclusion* relations on which monotonicity reasoning is based. It was then shown in Icard (2012) that the resulting system can be understood formally as an extension of the Monotonicity Calculus, with more type markings and correspondingly more classes of functions refining the classes of monotonic and antitonic functions.

A simple example of an inference that depends on exclusion is:

$$\textsf{Every porcupine is nocturnal} \ \Rightarrow\ \textsf{Not every porcupine is diurnal.}$$

Notice that nocturnal and diurnal do not stand in an inclusion relation, nor do every and not every. Thus, we cannot perform replacements as in the examples above. For that we need to know how the exclusion relations between such expressions are projected in different contexts.

The important observation, essentially made first in Keenan and Faltz (1984), is that type domains have more structure than an arbitrary pre-order. The relevant structure here is that of a *bounded distributive lattice*. This is a tuple $\mathbb{X} = (X, \vee, \wedge, \bot, \top)$, where X is a set, $\vee$ and $\wedge$ are commutative, associative, and idempotent operations which distribute over each other, and $\bot$ and $\top$ satisfy the identities $\bot \wedge x = \bot$, $\bot \vee x = x$, $\top \wedge x = x$, $\top \vee x = \top$. For instance, $\mathbb{D}_t$ is the smallest nontrivial bounded distributive lattice $\mathbb{2} = (\{0,1\}, +, \ \cdot\ , 0, 1)$; and the bounded distributive structure for predicates is the "powerset algebra" $(\wp(E), \cup, \cap, \emptyset, E)$. In fact, whenever $\mathbb{D}_\tau$ is a bounded distributive lattice, so is $\mathbb{D}_{\sigma\to\tau}$. All of this holds for the smaller class of "Boolean" lattices as well, but we do not need to make use of complements here.

Following MacCartney (2009) (as presented in Icard (2012)), we introduce the following set of relations, which are well-defined on elements of any bounded distributive lattice $\mathbb{X} = (X, \vee, \wedge, \bot, \top)$:

$$\begin{aligned} x \le y &\ :\ x \wedge y = x \\ x \ge y &\ :\ x \vee y = x \\ x | y &\ :\ x \wedge y = \bot \\ x \smile y &\ :\ x \vee y = \top \end{aligned}$$

We write $x \equiv y$ if both $x \le y$ and $x \ge y$; write $x \curlywedge y$ if both $x|y$ and $x \smile y$ (and we say that x and y are *mutually exclusive*); and write $x\#y$ for the universal (uninformative) relation on X. Thus we define the set $\mathcal{R}$ of relations to be: $\{\equiv, \le, \ge, \curlywedge, |, \smile, \#\}$.

Example 3. The predicates square and circular stand in the '|' relation since the set of square things and the set of circular things are disjoint.

In the space of quantifier meanings, we have more than five $\smile$ fewer than eight, since at least one of these holds of every two sets. Finally, animate stands in the ‘⅄’ relation to inanimate, since these are mutually exclusive and exhaustive.

Monotone functions preserve the basic inequalities $\leq$ and $\geq$, while antitone functions reverse them. Now that we have introduced several new relations, what classes of functions do we need to predict the relation between two complex expressions that differ only with respect to some subexpressions whose relation is known? That is, supposing uRv, what do we need to know about t in order to determine for which $R' \in \mathcal{R}$ we have $t[u]R't[v]$? It turns out, some familiar refinements of the classes of monotonic and antitonic functions are sufficient:

Definition 1. Suppose $f : \mathbb{X} \to \mathbb{Z}$ is a function on lattices.

> f is *additive* if $f(x \vee y) = f(x) \vee f(y)$.
> f is *multiplicative* if $f(x \wedge y) = f(x) \wedge f(y)$.
> f is *anti-additive* if $f(x \vee y) = f(x) \wedge f(y)$.
> f is *anti-multiplicative* if $f(x \wedge y) = f(x) \vee f(y)$.

That additivity and multiplicativity refine monotonicity follows from the easy observation that the following three conditions are equivalent: (a) f is monotonic; (b) $f(x) \vee f(y) \leq f(x \vee y)$; (c) $f(x \wedge y) \leq f(x) \wedge f(y)$. Similarly, anti-additivity and anti-multiplicativity refine antitonicity since the following three are equivalent: (a) f is antitonic; (b) $f(x \vee y) \leq f(x) \wedge f(y)$; (c) $f(x) \vee f(y) \leq f(x \wedge y)$.

It has been observed, at least since Zwarts (1981), that, for instance, no is anti-additive in both of its arguments, while every is anti-additive in its first argument and multiplicative in its second argument:

> No aardvark eats kelp or carrots $\equiv$
> No aardvark eats kelp and no aardvark eats carrots
>
> Every animal is a child and a grandchild $\equiv$
> Every animal is a child and every animal is a grandchild

In fact, all of these function classes are exemplified by expressions in English. We introduce a new set of type markings Σ extending $\mathcal{M}$:

$$\Sigma = \{\cdot, +, -, \diamondplus, \diamondminus, \boxplus, \boxminus, \oplus, \ominus\}.$$

Terms labeled with $\diamondplus$ will be additive; those with $\diamondminus$ anti-additive; $\boxplus$ corresponds to multiplicative; $\boxminus$ to anti-multiplicative; $\oplus$ corresponds to additive and multiplicative (for an expressions like *is*); finally, $\ominus$ is reserved for expressions that are anti-additive and anti-multiplicative, nearly amounting to outright negation.

The important observation for understanding exclusion-based inferences like those above is that these function classes "project" the relations in $\mathcal{R}$ in predictable ways. For instance, from the fact that nocturnal $\curlywedge$ diurnal and the fact that every is multiplicative, we can conclude:

Every porcupine is nocturnal $\mid$ Every porcupine is diurnal

If every were merely monotone in its second argument, this relation would not necessarily hold. Indeed, we can say explicitly what the "projectivity" behavior for each type of function is (MacCartney 2009). With $R \in \mathcal{R}$ and $\varphi \in \Sigma$, the *projection of* R *under* φ, written $[R]^{\varphi}$, is the strongest $R^* \in \mathcal{R}$ such that, whenever xRy and f is a φ-function, we must have $f(x)R^*f(y)$. The projectivity behavior of these function classes is summarized below.

[ ]	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\mid$	$\smile$
$+$	$\sqsubseteq$	$\sqsupseteq$	$\#$	$\#$	$\#$
$\diamondplus$	$\sqsubseteq$	$\sqsupseteq$	$\smile$	$\#$	$\smile$
$\boxplus$	$\sqsubseteq$	$\sqsupseteq$	$\mid$	$\mid$	$\#$
$\oplus$	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\mid$	$\smile$

[ ]	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\mid$	$\smile$
$-$	$\sqsupseteq$	$\sqsubseteq$	$\#$	$\#$	$\#$
$\diamondminus$	$\sqsupseteq$	$\sqsubseteq$	$\mid$	$\#$	$\mid$
$\boxminus$	$\sqsupseteq$	$\sqsubseteq$	$\smile$	$\smile$	$\#$
$\ominus$	$\sqsupseteq$	$\sqsubseteq$	$\curlywedge$	$\smile$	$\mid$

The final ingredient of exclusion reasoning is a *join* operation $\bowtie$, where $R \bowtie R'$ is understood to be the strongest relation $R^* \in \mathcal{R}$ such that whenever xRy and $yR'z$, it follows that xR^*z:

$\bowtie$	$\sqsubseteq$	$\sqsupseteq$	$\curlywedge$	$\mid$	$\smile$
$\sqsubseteq$	$\sqsubseteq$	$\#$	$\mid$	$\mid$	$\#$
$\sqsupseteq$	$\#$	$\sqsupseteq$	$\smile$	$\#$	$\smile$
$\curlywedge$	$\smile$	$\mid$	$\equiv$	$\sqsupseteq$	$\sqsubseteq$
$\mid$	$\#$	$\mid$	$\sqsubseteq$	$\#$	$\sqsubseteq$
$\smile$	$\smile$	$\#$	$\sqsupseteq$	$\sqsupseteq$	$\#$

We now have enough to finish deriving the example above, that No porcupine is diurnal follows from Every porcupine is nocturnal. We first observed that Every porcupine is nocturnal $\mid$ Every porcupine is diurnal. Moreover, we have every $\curlywedge$ not every, and in particular Every porcupine is diurnal $\curlywedge$ Not every porcupine is diurnal. Since the join $\mid \bowtie \curlywedge = \leq$, we can conclude Every porcupine is nocturnal $\leq$ Not every porcupine is diurnal. Summarizing in a natural-deduction-style proof, where $t =$ Every porcupine is diurnal, $s =$ Every porcupine is nocturnal, and $r =$ Not every porcupine is nocturnal:

$$\dfrac{\dfrac{\text{nocturnal} \curlywedge \text{diurnal}}{t|s} \qquad \dfrac{\text{every} \curlywedge \text{not every}}{s \curlywedge r}}{t \leq r} \; (\mid \bowtie \curlywedge = \leq)$$

To treat more complex cases that involve embeddings under multiple functional expressions, as in

> No porcupine that misses a warning sign is safe $\leq$
> Some porcupine that misses an important warning sign is in danger

we must understand the extension of the composition ($\circ$) operator from $\mathcal{M}$ to Σ which we saw in Section 2.4. The structure $(\Sigma, \circ)$ forms a monoid, with identity element $\oplus$ (Icard 2012). For example, from the fact that a is additive, misses is antitonic, and no is anti-additive and anti-multiplicative, we can conclude that No porcupine who misses a [ ] is safe is monotonic in [ ], since in fact $\ominus \circ - \circ \diamondplus = +$.

4 Applications

The systems and algorithms sketched above have found applications in psychology of language and computational linguistics. Before turning to formal foundations of the Monotonicity Calculus, we first offer an overview of some of the areas where these systems have proven useful.

4.1 Psychological

Monotonicty has been implicated in a number of psychological phenomena in language processing and language-based reasoning tasks. To take a simple example, consider the following sentence (inspired by an example in Geurts and van der Slik (2005)):

(a) Most Americans who know a foreign language speak it at home.

A typical speaker may not be able to judge under what conditions such a sentence will be true. Does it hold in a situation where most Americans who know two foreign languages speak only one of them at home? Or is it sufficient that most Americans know at least one of the languages they speak at home? Or need they speak most of the languages they know at home? Contrast this apparent underdeterminacy with the patent fact that the sentence in (a) entails that in (b).

(b) Most who know a foreign language speak it at home or at work.

In some sense, it does not matter which of the readings above is correct. This entailment holds on all of them. Relatedly, while there are significant logical differences between the quantifiers most and every, Oaksford and Chater (2001) have shown that inference patterns like

$$\frac{\textsf{most}\ X\ Y \qquad Y \leq Z}{\textsf{most}\ X\ Z} \qquad \text{and} \qquad \frac{\textsf{every}\ X\ Y \qquad Y \leq Z}{\textsf{every}\ X\ Z},$$

of which the inference from (a) to (b) is an instance, seem to be equally easy for subjects, despite the difference in logical complexity. One might take such evidence to suggest people sometimes are able to recognize entailments on the basis of these general monotonicity patterns.

Geurts (2003) has taken this idea further, demonstrating how a simple processing model based on a calculus closely related to what we outline here can explain many aspects of Oaksford and Chater's (1999) meta-analysis of syllogistic reasoning. For instance, among valid syllogisms those that require two applications of the monotonicity rule (our (Mono) below in Section 5.4) turn out to be more difficult for subjects than those that require only one, all other things equal. Generally, the results of this work are suggestive, if also preliminary.

One of the most intriguing aspects of monotonicty from a psycholinguistic point of view is the robust correspondence between antitone contexts and the syntactic distribution of a class of expressions called *negative polarity items* (NPIs). Perhaps the simplest generalization about these expressions—which include as examples in English at all, yet, any, a wink, . . .—is that they seem to appear almost solely in (locally) antitone contexts. For instance, while Everyone found any evidence is ungrammatical, No one found any evidence is perfectly grammatical. In fact, one of the intended uses of the internalized schema (Dowty 1994; Moss 2012) sketched above in Section 2.5 is to define categorial grammars that properly govern the syntactic distribution of NPIs. Thus, the monotonicity markings play a double role of licensing monotonicity inferences, as well as restricting which expressions will typically be recognized or generated by a grammar.

Within the class of NPIs, Zwarts (1981) and others have distinguished several subclasses of NPIs based on the strength of their preferred syntactic environments. Weak NPIs like any appear in arbitrary antitone contexts; strong NPIs, e.g., in years, require anti-additive contexts; while super-strong NPIs such as one bit require anti-additive and anti-multiplicative contexts. Thus, curiously, the correspondence between logical features and grammaticality extends to exclusion-based reasoning as well, in light of Section 3. Extending the internalized schema from monotonicity and antitonicity to internalized markings for exclusion relations, so as to govern the syntactic distribution of weak, strong, and super strong NPIs, is an interesting avenue for future work.

The exact nature of the generalizations about NPIs has been a matter of some controversy, as they also appear in questions and other contexts that are not antitone in any straightforward sense, e.g. Do you have any evidence?. Giannakidou (2011) contains an up-to-date summary of the data and theoretical proposals, as well as references to the literature on NPIs. There is some preliminary experimental work on this topic. Chemla et al. (2011), for example, suggest that speakers' perceived judgments of monotonicity may be better predictors of their grammaticality judgments than, say, the "true" logical facts about monotonicity in a given formalization. See also Szabolcsi et al. (2008) for experimental work on the link between antitonicity and negative polarity. It is an intriguing question why there should be such a close, if not perfect, correspondence at all between these logical features licensing monotonicity inferences and issues of which sentences are judged as well-formed. To our knowledge, this is still somewhat mysterious.

4.2 Computational

The computational problem of recognizing textual entailment (RTE)—that is, automatically determining which strings of text intuitively follow from which other strings of text—is an integral part of natural language processing. RTE is implicated in other critical natural language understanding tasks, including question answering, search, summarization, translation, and many others. The general RTE task is quite a difficult problem. Determining whether one claim follows from another can depend on just about any aspect of human knowledge, experience, and understanding. Just consider what might be required to recognize that The floor is very slippery follows from The floor is made of teflon and coated with motor oil. Other plausible entailments may be controversial to begin with: does it follow from Freedonia possesses enriched uranium that Freedonia is developing nuclear weapons? Many examples of this sort do show up in RTE contests and test suites, and certainly one would like to have an approach that works in open-ended domains. However, one of the intriguing observations from a logician's point of view is that a wide range of entailments follow distinct patterns—monotonicity being one of the most notable—and the basic world knowledge necessary may not go beyond simple lexical relations available from WordNet or some other lexical database. It seems reasonable to take advantage of these general "logical" patterns, that is, patterns validated on the basis of form alone, whenever possible.

MacCartney and Manning (2009) have developed an RTE system that includes monotonicity reasoning as the central component. Their "NatLog" system begins with some basic linguistic preprocessing: tokenization, parsing, named entity recognition, and so on. The system also runs a monotonicty marking algorithm like those outlined in Section 2, and builds on related work by Nairn et al. (2006) on so-called implicativity. In the end, NatLog makes a guess about the relation between the premise text and the hypothesis text using a sequence of edits bridging the two texts. Testing the system on the PASCAL RTE Challenge data (Dagan et al. 2005), NatLog outperformed the state-of-the-art Stanford RTE System on precision, though it fell far short on recall. The Stanford System is based on a maximum entropy classifier, which learns to make predictions from labeled text pairs using hand-coded features. Interestingly, a hybrid of the two systems outperformed both NatLog and the Stanford System on overall accuracy, suggesting that an integrated approach incorporating both statistical learning and logical reasoning may be desirable. The most detailed explanation of NatLog can be found in MacCartney (2009).

Given the importance and prevalence of monotonicity in ordinary reasoning, maintaining a list of expressions together with their monotonicity information promises to be useful. However, doing this manually may become quite cumbersome, particularly if we want to use the same basic algorithms and tools across multiple languages. Danescu et al. (2009) have taken a first step in addressing this by showing how antitone contexts in particular can be learned automatically. The trick is to capitalize on the close correspondence between antitone contexts and the distribution of NPIs, sketched above in Section 4.1. Intuitively, the more often an expression co-occurs with NPIs, the more likely it is to create antitone contexts. Using a list of well-established NPIs, Danescu et al. (2009) collect candidate expressions w by determining whether the following inequality holds, where $c_{NPI}(w)$ is the number of times w co-occurs with an NPI, $c(w)$ is the count of w in the corpus, and W is the lexicon:

$$\frac{c_{NPI}(w)}{\sum_{w' \in W} c_{NPI}(w')} > \frac{c(w)}{\sum_{w' \in W} c(w')}.$$

That is, the frequency of occurrences of w with NPIs should be greater than what we would expect from the frequency of w occurrences in the overall corpus. Their algorithm has good precision (80%), and most importantly, they discover a long list of antitone expressions that had not appeared on previous inventories. The algorithm has even been extended to achieve co-learning of antitone contexts and NPIs, e.g. for languages where extensive lists of NPIs are not already established (Danescu and Lee 2010). See also Cheung and Penn (2012) for related work. Note finally that in view of the connection between the inference patterns based on exclusion relations (Section 3) and subclasses of NPIs (Section 4.1), these methods could also be used to discover subclasses of antitone operators—anti-additive and anti-multiplicative—based on co-occurrence with weak, strong, and superstrong NPIs.

Generally, we believe that the formal investigation of logical and mathematical aspects of these systems for natural reasoning should be developed alongside these applied projects. Each stands to gain from insights the other can provide.

5 Formal Treatment

The centerpiece of this paper is a formal development of what we have seen. The material here is based on Icard and Moss (2013).

5.1 Types and Domains

Definition 2. As above, let $\mathcal{M} = \{+, -, \cdot\}$. We call $\mathcal{M}$ the set of *markings*, and we use m to denote an element of $\mathcal{M}$.

Definition 3. Let $\mathcal{B}$ be a set of *base types*. Working over some fixed set $\mathcal{B}$ and therefore suppressing mention of it, the full set of types $\mathcal{T}$ is defined as the smallest superset of $\mathcal{B}$, such that whenever $\sigma, \tau \in \mathcal{T}$, so is $\sigma \overset{m}{\to} \tau$, for each $m \in \mathcal{M}$.

Expressions of type $\sigma \overset{+}{\to} \tau$ will denote monotone functions, those of type $\sigma \overset{-}{\to} \tau$ antitone functions, and those of type $\sigma \overset{\cdot}{\to} \tau$ arbitrary functions. We therefore have a natural preorder on $\mathcal{M}$, whereby $m \sqsubseteq m'$ iff $m = m'$ or $m' = \cdot$. This ordering can be used to define a natural preorder on $\mathcal{T}$. Intuitively $\sigma \preceq \tau$ will mean that anything of type σ could also be considered as of type τ. So for function spaces, we take $\preceq$ to be "antitone in the domain argument and monotone in the codomain."

Example 4. In standard Montague semantics, we take $\mathcal{B}$ to be $\{e, t\}$. (However, recall from Section 2.6 that the work here extends to types for locations, times, sums, and other natural language categories with ordered domains.) In our example from algebra, we took it to be $\{r\}$.

Definition 4 ($\preceq$ on types)**.** Define $\preceq \in \mathcal{T} \times \mathcal{T}$ to be least such that $\tau \preceq \tau$, and whenever $\sigma' \preceq \sigma$ and $\tau \preceq \tau'$, and $m \sqsubseteq m'$, we have $\sigma \overset{m}{\to} \tau \preceq \sigma' \overset{m'}{\to} \tau'$.

Example 5. We return to the linguistic example, using base types e and t. We abbreviate $e \overset{\cdot}{\to} t$ by p (for "property"). A determiner (quantifier) such as every might be interpreted as an element of a marked type $p \overset{-}{\to} (p \overset{+}{\to} t)$. In some sense, this is the most specific type we could assign to every. But it could also be considered of type $p \overset{-}{\to} (p \overset{\cdot}{\to} t)$, for example, or even $p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$. Note that according to Def. 4,

$$p \overset{-}{\to} (p \overset{+}{\to} t) \preceq p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t).$$

The same holds for the type of some: $p \overset{+}{\to} (p \overset{+}{\to} t) \preceq p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$, and no: $p \overset{-}{\to} (p \overset{-}{\to} t) \preceq p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$.

Definition 5 ($\uparrow$ and $\vee$ on types)**.** We endow $\mathcal{M}$ with the obvious upper semilattice structure, writing $m_1 \vee m_2$ for m_1 if $m_1 = m_2$, and $\cdot$ otherwise. (Again, the dot $\cdot$ is one of the markings, hence an element of

$\mathcal{M}$.) $\uparrow$ is the smallest relation on types, and $\vee$ is the smallest function on types, with the properties that for all σ, τ_1, and τ_2:

1. $\sigma \uparrow \sigma$, and $\sigma \vee \sigma = \sigma$.
2. If $\tau_1 \uparrow \tau_2$, then $(\sigma \overset{m_1}{\to} \tau_1) \uparrow (\sigma \overset{m_2}{\to} \tau_2)$ for all $m_1, m_2 \in \mathcal{M}$, and
$$(\sigma \overset{m_1}{\to} \tau_1) \vee (\sigma \overset{m_2}{\to} \tau_2) \quad = \quad \sigma \overset{m_1 \vee m_2}{\longrightarrow} (\tau_1 \vee \tau_2).$$

We define $\sigma \mapsto \hat{\sigma}$ on $\mathcal{T}$ by $\hat{\sigma} = \sigma$ for σ basic, and $(\sigma \overset{m}{\to} \tau)\hat{} = \sigma \overset{\cdot}{\to} \hat{\tau}$.

Lemma 1. $\uparrow$ is an equivalence, and $\hat{\sigma}$ is the least upper bound in $\preceq$ of the (finite) $\uparrow$-equivalence class of σ.

As an ordered set, $(\mathcal{T}, \preceq)$ has some undesirable properties. For example, there are pairs of types that have incomparable upper bounds. These pathologies are largely "tamed" by Definition 5.

Example 6. Returning to Ex. 5, we have
$$p \overset{-}{\to} (p \overset{+}{\to} t) \ \uparrow \ p \overset{+}{\to} (p \overset{+}{\to} t) \ \uparrow \ p \overset{-}{\to} (p \overset{-}{\to} t).$$
The least upper bound for this $\uparrow$-equivalence class is $p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$. Intuitively, any expression of any of these types can just as well be considered an expression of type $p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$, the type of an arbitrary generalized quantifier.

Up until now, we have been dealing with the basics of the type system. We have yet to go into details on the syntax of higher-order terms. But before we do this, it will be informative to give the intended models for our languages. We call these *standard structures.*

Definition 6 (Structures)**.** A *standard structure* is a system $\mathcal{S} = \{\mathbb{D}_\tau\}_{\tau \in \mathcal{T}}$ of preorders (called *type domains*), one for each type $\tau \in \mathcal{T}$. We write $\mathbb{D}_\tau = (D_\tau, \leq_\tau)$ for the domain of type τ. For the base types $\beta \in \mathcal{B}$ there is no requirement on $\mathbb{D}_\beta$. For complex types $\sigma \overset{m}{\to} \tau$, we have several requirements:

1. $D_{\sigma \overset{+}{\to} \tau}$ is the set of all monotone functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$.
2. $D_{\sigma \overset{-}{\to} \tau}$ is the set of all antitone functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$.
3. $D_{\sigma \overset{\cdot}{\to} \tau}$ is the set of all functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$.
4. For all markings $m \in \mathcal{M}$, all types $\sigma, \tau \in \mathcal{T}$, and all $f, g \in D_{\sigma \overset{m}{\to} \tau}$, we have $f \leq_{\sigma \overset{m}{\to} \tau} g$ if and only if $f(a) \leq_\tau g(a)$ for all $a \in D_\sigma$. This is called the *pointwise* order.

Example 7. With $\mathcal{B} = \{e, t\}$, usually one takes $\mathbb{D}_e$ to be an arbitrary set, made into a *discrete preorder*: $x \leq y$ iff $x = y$. $\mathbb{D}_t$ is usually taken to be the two-element order $0 \leq 1$. Then we get a standard structure by defining $\mathbb{D}_\sigma$ by recursion on complex types. For example, $\mathbb{D}_{(e \overset{\cdot}{\to} t) \overset{\cdot}{\to} t}$ will be the set of *all* functions from $\mathbb{D}_{e \overset{\cdot}{\to} t}$ to $\mathbb{D}_t$, $\mathbb{D}_{(e \overset{\cdot}{\to} t) \overset{+}{\to} t}$ will be the set of monotone functions, and $\mathbb{D}_{(e \overset{\cdot}{\to} t) \overset{-}{\to} t}$ will be the set of antitone functions. In all cases, these are taken to be preorders using the pointwise order.

Continuing the algebra example, we take $\mathbb{D}_r$ to be $\mathbb{R} = (R, \leq)$, the real numbers with the usual order.

Incidentally, we speak of *standard structures* because one could instead interpret the language that we shall soon define on a more general class of structures (e.g., so called "Henkin models"). This is sometimes useful, but for our purposes in this paper the standard structures are sufficient, and the general definition is rather complicated.

We clearly have a natural embedding from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$ whenever $\sigma \preceq \tau$. This captures the sense in which anything of type σ could also be considered of type τ. When two types are related by $\uparrow$, their respective domains can both be embedded in a single domain. Since objects in this domain will be ordered, it will make sense to define ordering statements between expressions of $\uparrow$-related types in the formal language.

5.2 Language and Interpretation

Definition 7 (Unlabeled Typed Terms)**.** We begin the syntax with a set Con of *constants* together with a function $\mathsf{type} : \mathsf{Con} \to \mathcal{T}$. The set T of typed terms $t : \tau$ is defined recursively, as follows:

1. If $c \in \mathsf{Con}$, then $c : \mathsf{type}(c)$ is a typed term.
2. If $t : \sigma \overset{m}{\to} \tau$ and $u : \rho$ are typed terms and $\rho \preceq \sigma$, then $t(u) : \tau$ is a typed term.

Example 8. For an example pertaining to algebra, we take Con and type to be as given in (7), and with several more symbols

$$\begin{array}{l} \mathsf{abs} : r \overset{\cdot}{\to} r \\ \mathsf{0}, \mathsf{1}, \mathsf{2} : r \end{array}$$

Example 9. Here is a set of typed constants pertinent to natural language. We take plural nouns like cat, person, $\ldots : p$. Also, we take determiners (dets)

$$\begin{array}{l} \mathsf{every} : p \overset{-}{\to} (p \overset{+}{\to} t) \\ \mathsf{not\ every} : p \overset{+}{\to} (p \overset{-}{\to} t) \\ \mathsf{some} : p \overset{+}{\to} (p \overset{+}{\to} t) \\ \mathsf{no} : p \overset{-}{\to} (p \overset{-}{\to} t) \\ \mathsf{most} : p \overset{\cdot}{\to} (p \overset{+}{\to} t) \\ \mathsf{exactly}\ n : p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t) \end{array}$$

Transitive verbs like see could have type $(p \overset{+}{\to} t) \overset{+}{\to} t$. Then sentences of the form det+noun+verb+det+noun would correspond to terms of type t. (See Example 14.)

The typings of the determiners illustrate the use of this schema most clearly. They are reflections of the monotonicity phenomena that we have already seen. The observation that every is antitonic in its first argument and monotone in its second argument is exactly what the typing $p \xrightarrow{-} (p \xrightarrow{+} t)$ expresses.

A *term* is an object t such that there is a type τ with $t : \tau$. We assume that our notations arrange that every term has exactly one type. We interpret this language in a type domain as expected.

Definition 8 (Denotation)**.** For each term $t : \tau$, and for each $\tau' \succeq \tau$, we define $[\![t]\!]^{\mathcal{S}}_{\tau'}$ by induction on t.

1. The semantics begins with values $[\![c]\!]^{\mathcal{S}}_{\tau}$. We require that $[\![c]\!]^{\mathcal{S}}_{\tau}$ belong to $\mathbb{D}_{\tau}$.
2. If $t : \sigma \xrightarrow{m} \tau$ and $u : \sigma'$ with $\sigma' \preceq \sigma$, then $[\![t(u)]\!]^{\mathcal{S}}_{\tau} = [\![t]\!]^{\mathcal{S}}_{\sigma \xrightarrow{m} \tau}\big([\![u]\!]^{\mathcal{S}}_{\sigma'}\big)$.

In all cases, where $t : \tau \preceq \tau'$, we let $[\![t]\!]^{\mathcal{S}}_{\tau'} = [\![t]\!]^{\mathcal{S}}_{\tau}$.

Frequently we omit the superscript $\mathcal{S}$.

Example 10. Let $\mathcal{B} = \{r\}$, and $\mathcal{S}$ be the standard structure defined as follows. We take $\mathbb{D}_r$ to be $(\mathbb{R}, \leq)$. We take $[\![\mathsf{plus}]\!]$ to be the function from $\mathbb{R}$ to functions from $\mathbb{R}$ to itself which takes a real number a to the function $\lambda b.a + b$. For example, $[\![\mathsf{plus}]\!](67)(-3) = 64$. We have to check that $[\![\mathsf{plus}]\!]$ really belongs to $D_{r \xrightarrow{+} (r \xrightarrow{+} r)}$. This means: For each a, $[\![\mathsf{plus}]\!](a)$ is a monotone function: If $b \leq b'$, then $a+b \leq a+b'$. Moreover, the function from $\mathbb{R}$ to $(\mathbb{R} \xrightarrow{+} \mathbb{R})$ taking a to $[\![\mathsf{plus}]\!](a)$ is itself monotone. This means that if $a \leq a'$, then for all b, $a + b \leq a' + b$. We similarly use

$$\begin{array}{lcl@{\qquad}lcl}
[\![\mathsf{minus}]\!](a)(b) & = & a - b & [\![\mathsf{0}]\!] & = & 0 \\
[\![\mathsf{times}]\!](a)(b) & = & a \times b & [\![\mathsf{1}]\!] & = & 1 \\
[\![\mathsf{div2}]\!](a)(b) & = & a \div 2^b & [\![\mathsf{2}]\!] & = & 2 \\
[\![\mathsf{abs}]\!](a) & = & |a| & & &
\end{array}$$

It is now a fact of arithmetic that our semantics is appropriate in the sense that $[\![c]\!] \in \mathbb{D}_{\sigma}$ whenever $c : \sigma$ is part of the lexicon. That is, we have a *bona fide* semantics of all constants.

We then may work out the semantics of all terms. For example,

$$\begin{array}{rcl}
[\![\mathsf{plus\ 1\ 1}]\!] & = & [\![\mathsf{plus}]\!]([\![1]\!])([\![1]\!]) \\
 & = & [\![\mathsf{plus}]\!](1)(1) \\
 & = & 2.
\end{array}$$

5.3 Labeled terms

Because our typed terms may involve subterms within the scope of multiple functions, it is useful to *label* subterm occurrences to make clear what position that term is in. In fact, we make crucial use of this labeling in our proof system. For instance, if $t : \sigma \xrightarrow{-} \tau$ and $u : \rho \xrightarrow{-} \sigma$, then in $t(u(v)) : \tau$, subterm $v : \rho$ is in a monotone position. We have seen the simple algebra of markings $(\mathcal{M}, \circ)$ in Section 2.4, and the definition below captures the result of the monotonicity marking algorithms we outlined in Section 2.

Definition 9 (Labeled Terms)**.** Suppose u is a subterm occurrence in t. We shall find some $l \in \mathcal{M}$ which indicates the polarity of u inside t, and call it the *label* of the occurrence of u in t. We shall write this as $t[u^l]$. The definition is by recursion on terms:

1. If $u = t$, then $u[u^+]$; that is, the label of u is $+$ in u ;
2. If $s[u^l]$, then $s(v)[u^l]$; that is, subterms of a functor inherit their labels from the functor itself.
3. If $v[u^l]$ and $s : \tau \xrightarrow{m} \sigma$, then $s(v)[u^{m \circ l}]$; that is, an occurrence of u in an argument v of an application $s(v)$ has label $m \circ l$ in the overall term $s(v)$, where m is the label on the arrow in $\xrightarrow{m}$, and l is the label of the occurrence inside v.

Example 11. Let us compute the polarity of cat inside see every cat. The occurrence of cat inside itself is positive: cat[cat$^+$]. Recall that the type of every is of the form $\sigma \xrightarrow{-} \tau$. (Its type is $p \xrightarrow{-} (p \xrightarrow{+} t)$.) So the polarity of cat inside every cat is $- \circ + = -$. That is, every cat[cat$^-$]. Finally, a look at the type of verbs shows that see every cat[cat$^-$].

It is also sensible to shorten the notation a bit and write every [cat$^-$] and see every [cat$^-$]. In the same way, we would have see some [cat$^+$] and see most [cat$^\cdot$].

Example 12. We have already seen several different definitions of the polarity operation. In particular, Example 6 shows that

$$\textsf{div2 minus } [x^+]\ [y^-] \textsf{ minus } [z^-] \textsf{ plus } [v^+]\ [w^+].$$

Lemma 2 (Soundness of Labeling Scheme)**.** Suppose $t : \tau$ and $u : \rho$ is a subterm occurrence of t such that $t[u^l]$ with subterm u labeled by $l \in \{+, -\}$. Then for any structure $\mathcal{S}$, supposing $[\![u]\!]_{\hat{\rho}} \leq_{\hat{\rho}} [\![v]\!]_{\hat{\rho}}$:

1. If $l = +$, then $[\![t]\!]_\tau \leq_\tau [\![t^{v \leftarrow u}]\!]_\tau$;
2. If $l = -$, then $[\![t^{v \leftarrow u}]\!]_\tau \leq_\tau [\![t]\!]_\tau$.

Here $t^{v \leftarrow u}$ is the term that results from substituting v for the occurrence of u in t. In other words, if a subterm occurrence is labeled by $+$ $(-)$, it is indeed in a monotone (antitone) position. This is the crucial point about the correspondence between polarity and monotonicity.

5.4 Monotonicity Calculus

We are interested in proof relations between sets of inequalities Γ and individual inequality statements $s \leq t$. As discussed above, we allow inequality statements between terms whose types are $\uparrow$-related, since these are exactly the terms that should be $\leq$-comparable semantically.

Definition 10 (Satisfaction)**.** Where $s : \sigma$ and $t : \tau$, and if $\sigma \uparrow \tau$, we write $\mathcal{S} \models s \leq t$ if $[\![s]\!]_{\hat{\sigma}} \leq_{\hat{\sigma}} [\![t]\!]_{\hat{\sigma}}$. We shall always use Γ to denote a set of statements of the form $u \leq v$. We write $\Gamma \models s \leq t$ if, whenever $\mathcal{S} \models u \leq v$ for all statements $u \leq v \in \Gamma$, also $\mathcal{S} \models s \leq t$.

Definition 11 (Monotonicity Calculus)**.** The Monotonicity Calculus is given by the following rules:

$$(\textsc{Refl})\ \frac{}{t \leq t} \qquad (\textsc{Trans})\ \frac{t \leq u \qquad u \leq v}{t \leq v}$$

$$(\textsc{Mono})\ \frac{u \leq v}{t[u^+] \leq t^{v \leftarrow u}} \qquad (\textsc{Anti})\ \frac{u \leq v}{t^{v \leftarrow u} \leq t[u^-]}$$

$$(\textsc{Point})\ \frac{s \leq t}{s(u) \leq t(u)}$$

If Γ is a set of statements of the form $u \leq v$, we say $\Gamma \vdash s \leq t$ if $s \leq t \in \Gamma$, or there is a proof of $s \leq t$ from inequalities in Γ using the rules above.

Example 13. It might be amusing to see that some facts of arithmetic can now be derived from assumptions. For example,

$$\{\mathsf{0} \leq \mathsf{1}, \mathsf{1} \leq \mathsf{2}\} \vdash \mathsf{minus}\ \mathsf{1}\ \mathsf{1} \leq \mathsf{minus}\ \mathsf{2}\ \mathsf{0}.$$

That is, if we assume (the facts about numbers) that $0 \leq 1$ and $1 \leq 2$, and if we work in our logic, and in particular if we assume the typing of minus that we have seen, then we can prove that $1 - 1 \leq 2 - 0$:

$$\dfrac{\dfrac{\dfrac{\mathsf{1} \leq \mathsf{2}}{\mathsf{minus}\ [\mathsf{1}^+] \leq \mathsf{minus}\ \mathsf{2}}(\textsc{Mono})}{\mathsf{minus}\ \mathsf{1}\ \mathsf{1} \leq \mathsf{minus}\ \mathsf{2}\ \mathsf{1}}(\textsc{Point}) \qquad \dfrac{\mathsf{0} \leq \mathsf{1}}{\mathsf{minus}\ \mathsf{2}\ [\mathsf{1}^-] \leq \mathsf{minus}\ \mathsf{2}\ \mathsf{0}}(\textsc{Anti})}{\mathsf{minus}\ \mathsf{1}\ \mathsf{1} \leq \mathsf{minus}\ \mathsf{2}\ \mathsf{0}}(\textsc{Trans})$$

We need our set of assumptions $\Gamma = \{\mathsf{0} \leq \mathsf{1}, \mathsf{1} \leq \mathsf{2}\}$ because the assertions in it are not true in all structures that we could conceivably use to interpret the language. The point of the calculus is not to tell us specific facts about numbers but rather to allow us to infer generalities, assertions which hold in all models of some set of assumptions. This is the standard account of model-theoretic consequence used in semantics.

Example 14. Let Γ contain $\textsf{every} : p \xrightarrow{-} (p \xrightarrow{+} t) \leq \textsf{most} : p \xrightarrow{\cdot} (p \xrightarrow{+} t)$, $\textsf{cat} : p \leq \textsf{animal} : p$, and $\textsf{child} : p \leq \textsf{person} : p$. Below is a small derivation in the calculus:

$$\dfrac{\dfrac{\textsf{cat} \leq \textsf{animal}}{\textsf{every } [\textsf{animal}^-] \leq \textsf{every cat}}\,(\textsc{Anti}) \qquad \dfrac{\textsf{every} \leq \textsf{most}}{\textsf{every cat} \leq \textsf{most cat}}\,(\textsc{Point})}{\dfrac{\textsf{every animal} \leq \textsf{most cat}}{\textsf{every animal runs} \leq \textsf{most cat runs}}\,(\textsc{Point})}\,(\textsc{Trans})$$

Example 15. We can also fruitfully combine the numerical example with the linguistic one. We take the set $\mathcal{B}$ of base types to be $\{e, t, r\}$. We use all the syntax which we have already seen, and also

$$\begin{aligned}
&\textsf{at least} : r \xrightarrow{-} (p \xrightarrow{+} (p \xrightarrow{+} t))\\
&\textsf{at most} : r \xrightarrow{+} (p \xrightarrow{-} (p \xrightarrow{-} t))\\
&\textsf{more than} : r \xrightarrow{-} (p \xrightarrow{+} (p \xrightarrow{+} t))\\
&\textsf{less than} : r \xrightarrow{+} (p \xrightarrow{-} (p \xrightarrow{-} t)).
\end{aligned}$$

Then the natural set Γ of assumptions would include

$$\begin{aligned}
&\textsf{0} \leq \textsf{1}, \textsf{1} \leq \textsf{2}, \ldots, \quad \textsf{more than} \leq \textsf{at least}, \quad \textsf{less than} \leq \textsf{at most},\\
&\textsf{some} \leq \textsf{at least 1}, \quad \textsf{at least 1} \leq \textsf{some}
\end{aligned}$$

For example, we could prove

$$\textsf{more than three people walk} \leq \textsf{at least two people walk}.$$

All three of the above examples use only unembedded versions of the (Mono) and (Anti) rules. That is, we could have stated these rules as follows, where $t : \sigma \xrightarrow{+} \tau$, $s : \sigma \xrightarrow{-} \tau$, and $u, v : \sigma' \preceq \sigma$.

$$(\textsc{Mono}^*)\ \frac{u \leq v}{t(u) \leq t(v)} \qquad (\textsc{Anti}^*)\ \frac{u \leq v}{s(v) \leq s(u)}$$

It is then easily shown that the more common presentations of the rules, viz. (Mono) and (Anti), can be derived from the system with (Mono*) and (Anti*) instead. It is an interesting question, which of these is more psychologically natural or computationally sensible: deriving labelings for terms as in Section 5.3 above, or restricting application of the monotonicity and antitonicity rules to only atomic function symbols? We leave this as an open question for future work.

5.5 Soundness and Completeness

A natural question about any axiomatic system is whether it is *sound* and *complete* with respect to an intended class of interpretations. Soundness means that every inequality which is proved in the calculus holds in all of the interpretations we are considering, no matter what the meanings of the individual lexical items. Completeness is the converse; it says that the calculus is strong enough to derive (from a set Γ of assumptions) all of the inequalities true in all models of Γ.

Theorem 1 (Soundness of the Monotonicity Calculus)**.** If $\Gamma \vdash s \leq t$, then $\Gamma \models s \leq t$. (For the proof see Icard and Moss (2013).)

Completeness—the statement that $\Gamma \models s \leq t$ implies $\Gamma \vdash s \leq t$—we have proven in some special cases (Icard and Moss 2013). For instance, when we consider a wider class of structures than those defined above in Def. 6, the result holds. We also conjecture that it holds for this smaller class of standard structures based on hierarchies of functions, as defined here.

We also should point out that in many applications, one is not interested in *truth in all possible models* but rather in truth in all "reasonable" models. For instance, in the linguistic example, when we think of the inferential patterns of determiners, many important patterns go beyond monotonicity (and even the extension to reasoning about exclusion relations). We did not incorporate de Morgan's laws, so we will not be able to infer someone does not cry from not everyone cries. To mention another source of incompleteness, from Pat is a clarinetist, it follows that everyone who likes every clarinetist likes Pat, and even everyone who likes everyone who likes Pat likes everyone who likes every clarinetist. Our calculus as it stands is not strong enough to derive these. Furthermore, we have no way to use variables. For example, in the algebra example, we might like to include a numerical variable $x : r$ and take Γ to contain $x \leq \mathsf{abs}(x)$. Then we could deduce things like

$$\mathsf{minus}\ 0\ 1 \quad \leq \quad \mathsf{abs}\ \mathsf{minus}\ 0\ 1.$$

Indeed, there are some linguistic applications involving introduction rules and "hypothetical reasoning" in extended categorial grammars, which make crucial use of variables and lambda abstraction.[3] We leave such extensions of the formalism developed here for future work.

6 Conclusion

Monotonicity is an important concept in the study of logic and language. We hope to have shown some of the reasons for this, both from a linguistic point of view and from a logical point of view. There are certainly many extensions to pursue beyond what we have discussed, both regarding psycholinguistic aspects of monotonicty and polarity, and regarding more expressive logical systems.

[3]See Zamansky et al. (2006) for a natural logic system similar to what we present here, but based on Lambek Calculus.

From the point of view of language technology, in Section 4 we described some of the exciting recent work applying ideas and algorithms from logic, viz. Monotonicity Calculus, to practical tasks in natural language processing, particularly for the problem of recognizing textual entailment. In turn, some of the logical work we described, e.g. in Section 3, was inspired by practical work in computational linguistics. It is our hope, and our prediction, that this trend of mutual influence will continue well into the future.

References

Chater, Nick and Mike Oaksford. 1999. The probability heuristics model of syllogistic reasoning. *Cognitive Psychology* 38:191–258.

Chemla, Emmanuel, Vincent Homer, and Daniel Rothschild. 2011. Modularity and intuitions in formal semantics: the case of polarity items. *Linguistics and Philosophy* 34(6):537–570.

Cheung, Jackie and Gerald Penn. 2012. Unsupervised detection of downward-entailing operators by maximizing classification certainty. In *13th Conference of the European Chapter of the Association for Computational Linguistics*.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Danescu, Cristian and Lillian Lee. 2010. Don't 'have a clue'? Unsupervised co-learning of downward-entailing operators. In *Proceedings of ACL*.

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

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

Geurts, Bart. 2003. Reasoning with quantifiers. *Cognition* 86(3):223–251.

Geurts, Bart and Frans van der Slik. 2005. Monotonicity and processing load. *Journal of Semantics* 22.

Giannakidou, Anastasia. 2011. Negative and positive polarity items. In C. Maienborn, K. von Heusinger, and P. Portner, eds., *Semantics: An International Handbook of Natural Language Meaning*. Wouter de Gruyter.

Icard, Thomas F. 2012. Inclusion and exclusion in natural language. *Studia Logica* 100(4):705–725.

Icard, Thomas F. and Lawrence S. Moss. 2013. A complete calculus of monotone and antitone higher-order functions. Unpublished ms.

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

MacCartney, Bill. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

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

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

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

Oaksford, Mike and Nick Chater. 2001. The probabilistic approach to human reasoning. *Trends in Cognitive Sciences* 5:349–357.

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

Szabolcsi, Anna, Lewis Bott, and Brian McElree. 2008. The effect of negative polarity items on inference verification. *Journal of Semantics* 25(4):411–450.

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

van Benthem, Johan. 1991. *Language in Action: Categories, Lambdas, and Dynamic Logic*, vol. 130 of *Studies in Logic*. Elsevier, Amsterdam.

van Benthem, Johan. 2008. A brief history of natural logic. In M. N. M. M. Chakraborty, B. Löwe and S. Sarukkai, eds., *Logic, Navya-Nyaya and Applications, Homage to Bimal Krishna Matilal*. London: College Publications.

van Eijck, Jan. 2007. Natural logic for natural language. In B. ten Cate and H. Zeevat, eds., *6th International Tbilisi Symposium on Logic, Language, and Computation*. Springer.

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

Zwarts, Frans. 1981. Negatief polaire uitdrukkingen I. In *GLOT*, vol. 4, pages 35–132.