Lawrence S. Moss

The Soundness of Internalized Polarity Marking

Abstract. This paper provides a foundation for the polarity marking technique introduced by David Dowty [3] in connection with monotonicity reasoning in natural language and in linguistic analyses of negative polarity items based on categorial grammar. Dowty's work is an alternative to the better-known algorithmic approach first proposed by Johan van Benthem [11], and elaborated by Víctor Sánchez Valencia [10]. Dowty's system *internalized* the monotonicity/polarity markings by generating strings using a categorial grammar whose categories already contain the markings that the earlier system would obtain by separate steps working on an already-derived string. Despite the linguistic advantages of the internalized system, no soundness proof has yet been given for it. This paper offers an account. The leading mathematical idea is to interpret categorial types as preorders (in order to talk about monotonicity in the first place), and then to add a primitive operation to the type hierarchy of taking the opposite of a preorder (in order to capture monotone decreasing functions). At the same time, the use of internalized categories also raises issues. Although these will not be addressed in full, the paper points out possible approaches to them.

1. Introduction

One enterprise at the border of logic and linguistics is the crafting of logical systems which capture aspects of inference in natural language. The ultimate goal is a logical system which is strong enough to represent the most common inferential phenomena in language and yet is weak enough to be decidable. Another goal would be to use representations which are as close as possible to the surface forms. This second goal is likely to be unobtainable, and when one backs off to consider structured representations, it is natural to turn to those suggested by linguistic formalisms of one form or another.

The particular strand of this enterprise that concerns us in this paper is the *natural logic* program initiated by Johan van Benthem [11, 12], and elaborated in Víctor Sánchez Valencia [10]. As mentioned in van Benthem [13], the "proposed ingredients" of a logical system to satisfy the goals would consist of several "modules":

Presented by **Name of Editor**; *Received* December 1, 2005

(a) Monotonicity Reasoning, i.e., Predicate Replacement,

(b) Conservativity, i.e., Predicate Restriction, and also

(c) Algebraic Laws for inferential features of specific lexical items.

We are only concerned with (a) in this paper, and we are mainly concerned with an alternative to van Benthem's seminal proposal.

There are several reasons why monotonicity is worthy of study on its own: it is arguably tied up with linguistic phenomena, it is a pervasive phenomenon in actual reasoning, and finds its way into computational systems for linguistic processing (cf. Nairn, Condoravdi and Karttunen [8] and MacCartney and Manning [6]). The leading idea in van Benthem's work is to develop a formal approach to monotonicity on top of categorial grammar (CG). This makes sense because the semantics of CG is given in terms of functions, and monotonicity in the semantic sense is all about functions. The success story of the natural logic program is given by the *monotonicity calculus*, a way of determining the polarity of words in a given sentence. Before turning to it, we should explain the difference between monotonicity and polarity. We follow the usage of Bernardi [1], Section 4.1:

> The differences between monotonicity and polarity could be summarized in a few words by saying that monotonicity is a property of functions .... On the other hand, polarity is a static syntactic notion which can be computed for all positions in a given formula. This connection between the semantic notion of monotonicity and the syntactic one of polarity is what one needs to reach a proof theoretical account of natural reasoning and build a natural logic.

This point might be clarified with an example of the monotonicity calculus. Consider the sentence *No dog chased every cat*. The sentence is to be derived in some categorial grammar, and to be brief we suppress all the details on this except to say that a derivation would look like the first two trees in Figure 1, except without the + and − marks. Given the derivation, one would first add some notations to indicate that the interpretation of *every*, as a function from noun meanings, is monotone decreasing but results in a function that is monotone increasing. Also, the interpretation of *no*, as a function from noun meanings, is monotone decreasing and again results in a function that is monotone decreasing.

In the topmost derivation, the tree is decorated by monotonicity markings which were propagated from the information about *every* and *no* according to a rule which need not concern us. One then turns the Monotonicity

$$\dfrac{\dfrac{\mathsf{no}^{+} : (pr,(pr,t)) \quad \mathsf{dog}^{-} : pr}{\mathsf{no}(\mathsf{dog})^{+} : (pr,t)} \quad \dfrac{\mathsf{chase}^{+} : ((pr,t),pr) \quad \dfrac{\mathsf{every}^{+} : (pr,(pr,t)) \quad \mathsf{cat}^{-} : pr}{\mathsf{every}(\mathsf{cat})^{+} : (pr,t)}}{\mathsf{chase}(\mathsf{every}(\mathsf{cat}))^{-} : pr}}{\mathsf{no}(\mathsf{dog})(\mathsf{chase}(\mathsf{every}(\mathsf{cat})))^{+} : t}$$

Monotonicity Marking

$$\dfrac{\dfrac{\mathsf{no}^{+} : (pr,(pr,t)) \quad \mathsf{dog}^{-} : pr}{\mathsf{no}(\mathsf{dog})^{+} : (pr,t)} \quad \dfrac{\mathsf{chase}^{+} : ((pr,t),pr) \quad \dfrac{\mathsf{every}^{-} : (pr,(pr,t)) \quad \mathsf{cat}^{+} : pr}{\mathsf{every}(\mathsf{cat})^{-} : (pr,t)}}{\mathsf{chase}(\mathsf{every}(\mathsf{cat}))^{-} : pr}}{\mathsf{no}(\mathsf{dog})(\mathsf{chase}(\mathsf{every}(\mathsf{cat})))^{+} : t}$$

Polarity Determination

$$\dfrac{\dfrac{\mathsf{no}^{+} : (-pr,(-pr,t)) \quad \mathsf{dog}^{-} : -pr}{\mathsf{no}^{+}(\mathsf{dog}^{-}) : (-pr,t)} \quad \dfrac{\mathsf{chase}_1^{-} : ((-pr,-t),-pr) \quad \dfrac{\mathsf{every}^{-} : (pr,(-pr,-t)) \quad \mathsf{cat}^{+} : pr}{\mathsf{every}^{-}(\mathsf{cat}^{+}) : (-pr,-t)}}{\mathsf{chase}_1^{-}(\mathsf{every}^{-}(\mathsf{cat}^{+})) : -pr}}{\mathsf{no}^{+}(\mathsf{dog}^{-})(\mathsf{chase}_1^{-}(\mathsf{every}^{-}(\mathsf{cat}^{+}))) : t}$$

The Fully Internalized Tree

Figure 1. The first two derivations illustrate van Benthem's monotonicity calculus [11, 12]; they are discussed in Section 1. The final derivation shows the internalized polarity marking scheme as proposed by Dowty [3] and as analyzed in this paper.

Marking tree into the Polarity Determination tree, the second tree on the page. The rule is: for each node, count the number of $-$ signs from it to the root of the tree (at the bottom). Then put a $+$ on the node if the count is an even number, and $-$ if the count is odd. This changes the signs for *every*, *cat*, and *every cat*. The resulting signs are the polarities. For example, there is a $+$ marking on *cat*. This indicates that this occurrence of *cat* might be replaced by a word with a "larger" value. For example, replacing *cat* with *animal* gives a valid inference: *No dog chased every cat* implies *No dog chased every animal* [1]. On the other hand, *dog* is marked $-$, corresponding to the fact that replacement inferences with it go in the opposite direction. So replacing *dog* with *animal* would not give a validity; instead, replace with the "smaller" phrase *old dog*.

As Dowty [3] mentions, "The goal [in his paper] is to 'collapse' the independent steps of Monotonicity Marking and Polarity Determination into the syntactic derivation itself, so that words and constituents are generated with the markings already in place that they would receive in Sánchez' polarity summaries." His motivation is mainly linguistic rather than logical: "the format of Sánchez' system ... is no doubt highly appropriate for the logical studies he developed it for, it is unsuitable for the linguistics applications I am interested in." For this reason, he gave an "alternative formulation." This system is "almost certainly equivalent to Sánchez'," but he lacked a proof. Bernardi [1] also notes that while her system and Sánchez' "are proven to be sound, no analogous result is known for" the system proposed by Dowty. However her presentation makes it clear that there are advantages to the internalization of polarity markers. Others who use a system inspired by [3] include Christodoulopoulos [2].

The purpose of this short paper is to offer a soundness proof for internalized monotonicity markings, and to discuss related issues. The first of these issues has to do with a choice in the overall categorial architecture. We wish to take seriously the idea that syntactic categories might be interpreted in ordered settings. Given two types σ and τ, we want to consider *two* resulting types: the *upward monotone functions* and the *downward monotone functions*. The first alternative is to generate types as follows: begin with a set $\mathcal{T}_0$ of basic types, and then build the smallest set $\mathcal{T}_1 \supseteq \mathcal{T}_0$ closed in the following way:

If $\sigma, \tau \in \mathcal{T}_1$, then also $(\sigma, \tau)^+ \in \mathcal{T}_1$ and $(\sigma, \tau)^- \in \mathcal{T}_1$.

[1] Incidentally, we are taking this sentence and others like it in the subject wide-scope reading: there is no dog with the property that it chases every cat.

This would seem to be the most straightforward way to proceed, and it actually seems to be close to the way that is presented in the literature[2]. Indeed, we worked out the theory this way before deciding to rewrite this paper based on a second alternative. This second way is based on the observation that a downward monotone function from a preorder $\mathbb{P}$ to a preorder $\mathbb{Q}$ is the same thing as an *upward* monotone function from $\mathbb{P}$ to *the opposite order* $-\mathbb{Q}$. This order $-\mathbb{Q}$ has the same points as $\mathbb{Q}$, but the order is "upside down." The suggestion is that we generate types a little differently. We take smallest set $\mathcal{T}_1 \supseteq \mathcal{T}_0$ closed in the following way:

If $\sigma, \tau \in \mathcal{T}_1$, then also $(\sigma, \tau) \in \mathcal{T}_1$.

If $\sigma \in \mathcal{T}_1$, then also $-\sigma \in \mathcal{T}_1$.

We shall call this the *fully internalized* scheme. This includes copies of the types done the first way, by viewing $(\sigma, \tau)^+$ as (σ, τ) and $(\sigma, \tau)^-$ as $(\sigma, -\tau)$. It is a strictly bigger set, since $-\sigma$ for the basic types enters into the fully internalized hierarchy but not the lesser one.

2. Background: Preorders and their Opposites

For the work in this paper we need to use *preorders*. Our goal in this section is to state the facts which are needed in as compact a manner as possible, and also to mention a few examples that will be used in the sequel.

A preorder is a pair $\mathbb{P} = (P, \leq)$ consisting of a set P together with a relation $\leq$ which is *reflexive* and *transitive*. This means that the following hold:

1. $p \leq p$ for all $p \in P$.
2. If $p \leq q$ and $q \leq r$, then $p \leq r$.

Frequently one assumes that the order has additional properties: it may be also anti-symmetric, or a lattice order, etc. For most of what we are doing, no extra assumptions are needed. Indeed, the specific results do not use the preorder properties very much, and some of the work in this section goes through for absolutely arbitrary relations. However, since the intended application is to a setting in which the relation represents some kind of inference, it makes sense to state the results for the weakest class

[2]For example, Dowty [3] states his category formation rules in this way, adding directional versions. But shortly thereafter he mentions that lexical items will "be entered in two categories." Adding downward-marked versions of the base categories is tantamount to what we call the fully internalized scheme.

of mathematical objects that has something that looks like "inference," and this seems to be preorders.

EXAMPLE 2.1. For any set X, we have a preorder $\mathbb{X} = (X, \leq)$, where $x \leq y$ iff $x = y$. This is called the *flat preorder* on X. More interestingly, for any set X we have a preorder $\mathcal{P}(X)$ whose set part is the set of subsets of X, and where $p \leq q$ iff p is a subset of q. Another preorder is $\mathbb{2} = \{\mathsf{F}, \mathsf{T}\}$ with $\mathsf{F} < \mathsf{T}$.

The natural class of maps between preorders $\mathbb{P}$ and $\mathbb{Q}$ is the set of *monotone functions*: the functions $f : P \to Q$ with the property that if $p \leq q$ in $\mathbb{P}$, then $f(p) \leq f(q)$ in $\mathbb{Q}$. *When we write $f : \mathbb{P} \to \mathbb{Q}$ in this paper, we mean that f is monotone.* We write $[\mathbb{P}, \mathbb{Q}]$ for the set of monotone functions from $\mathbb{P}$ to $\mathbb{Q}$. $[\mathbb{P}, \mathbb{Q}]$ is itself a preorder, with the *pointwise order*:

$$f \leq g \text{ in } [\mathbb{P}, \mathbb{Q}] \quad \text{iff} \quad \text{for all } p \in P,\ f(p) \leq g(p) \text{ in } \mathbb{Q}$$

Let $\mathbb{P}$ and $\mathbb{Q}$ be preorders. The *product preorder* $\mathbb{P} \times \mathbb{Q}$ is the preorder whose universe is the cartesian product $P \times Q$, and with $(p, q) \leq (p', q')$ iff $p \leq p'$ in $\mathbb{P}$ and $q \leq q'$ in $\mathbb{Q}$.

Preorders $\mathbb{P}$ and $\mathbb{Q}$ are *isomorphic* if there is a function $f : P \to Q$ which is bijective (one-to-one and maps P onto Q) and also has the property that $p \leq p'$ iff $f(p) \leq f(p')$. We write $\mathbb{P} \cong \mathbb{Q}$. We also identify isomorphic preorders, and so we even write $\mathbb{P} = \mathbb{Q}$ to make this point.

PROPOSITION 2.2. *For all preorders $\mathbb{P}$, $\mathbb{Q}$, and $\mathbb{R}$, and all sets X:*

1. *For each $p \in P$, the function $app_p : [\mathbb{P}, \mathbb{Q}] \to Q$ given by $app_p(f) = f(p)$ is an element of $[[\mathbb{P}, \mathbb{Q}], \mathbb{Q}]$.*
2. $[\mathbb{P} \times \mathbb{Q}, \mathbb{R}] \cong [\mathbb{P}, [\mathbb{Q}, \mathbb{R}]]$.
3. $[\mathbb{X}, \mathbb{2}] \cong \mathcal{P}(X)$.

PROOF. For part (1), let $f \leq g$ in $[\mathbb{P}, \mathbb{Q}]$. Then $app_p(f) = f(p) \leq g(p) = app_p(g)$.

In the second part, let $f : [\mathbb{P} \times \mathbb{Q}, \mathbb{R}] \to [\mathbb{P}, [\mathbb{Q}, \mathbb{R}]]$ be $f(a)(p)(q) = a(p, q)$. It is routine to check that f is an isomorphism.

In part (3), the isomorphism is $\varphi : [\mathbb{X}, \mathbb{2}] \to \mathcal{P}(X)$ given by $\varphi(f) = \{p \in P : f(p) = \mathsf{T}\}$. ∎

Antitone functions and opposites of preorders We are also going to be interested in *antitone functions* from a preorder $\mathbb{P}$ to a preorder $\mathbb{Q}$. These are the functions $f : P \to Q$ with the property that if $p \leq q$ in $\mathbb{P}$, then $f(q) \leq f(p)$ in $\mathbb{Q}$. We can express things in a more elegant way using the concept of the *opposite preorder* $-\mathbb{P}$ of a given preorder $\mathbb{P}$. This is the preorder with the same set part, but with the opposite order. Formally,

$$\begin{array}{lll} -P & = & P \\ p \leq q \text{ in } -\mathbb{P} & \text{iff} & q \leq p \text{ in } \mathbb{P} \end{array}$$

Example 2.3. One example is the negation operation on truth values: $\neg : \mathbb{2} \to -\mathbb{2}$. For another, let $\mathbb{P}$ be any preorder, and let $\mathbb{Q}$ be $\mathcal{P}(P)$, the power set preorder on the underlying set P of $\mathbb{P}$. Let $\uparrow\colon P \to -Q$ be defined by $\uparrow(p) = \{p' \in P : p \leq p' \text{ in } \mathbb{P}\}$. Since $p \leq p'$ in $\mathbb{P}$ implies $\uparrow(p) \supseteq \uparrow(p')$, we indeed see that $\uparrow\colon \mathbb{P} \to -\mathbb{Q}$.

We collect the most important facts on this operation below.

Proposition 2.4. *For all preorders $\mathbb{P}$, $\mathbb{Q}$, and $\mathbb{R}$, and all sets X:*

1. $-(-\mathbb{P}) = \mathbb{P}$.
2. $[\mathbb{P}, -\mathbb{Q}] = -[-\mathbb{P}, \mathbb{Q}]$.
3. $[-\mathbb{P}, -\mathbb{Q}] = -[\mathbb{P}, \mathbb{Q}]$.
4. *If $f : \mathbb{P} \to \mathbb{Q}$ and $g : \mathbb{Q} \to \mathbb{R}$, then $g \cdot f : \mathbb{P} \to \mathbb{R}$.*
5. *If $f : \mathbb{P} \to -\mathbb{Q}$ and $g : \mathbb{Q} \to -\mathbb{R}$, then $g \cdot f : \mathbb{P} \to \mathbb{R}$.*
6. $-(\mathbb{P} \times \mathbb{Q}) \cong -\mathbb{P} \times -\mathbb{Q}$.
7. $\mathbb{X} \cong -\mathbb{X}$.
8. $-[\mathbb{X}, \mathbb{P}] \cong [\mathbb{X}, -\mathbb{P}]$.

Proof. For part (1), recall first that $\mathbb{P}$ and $-\mathbb{P}$ have the same underlying set. It follows that $\mathbb{P}$ and $-(-\mathbb{P})$ also have this the same underlying set. Concerning the order: $x \leq y$ in $\mathbb{P}$ iff $y \leq x$ in $-\mathbb{P}$ iff $x \leq y$ in $-(-\mathbb{P})$.

For (2), suppose that $f : \mathbb{P} \to -\mathbb{Q}$. Then f is a set function from P to Q. To see that f is a monotone function from $-\mathbb{P}$ to $\mathbb{Q}$, let $x \leq y$ in $-\mathbb{P}$. Then $y \leq x$ in $\mathbb{P}$. So $f(y) \leq f(x)$ in $-\mathbb{Q}$. And thus $f(x) \leq f(y)$ in $\mathbb{Q}$, as desired. This verifies that as sets, $[\mathbb{P}, -\mathbb{Q}] = [-\mathbb{P}, \mathbb{Q}]$. To show that as preorders $[\mathbb{P}, -\mathbb{Q}] = -[-\mathbb{P}, \mathbb{Q}]$, we must check that if $f \leq g$ in the order on $[\mathbb{P}, -\mathbb{Q}]$, then $g \leq f$ in the order on $[-\mathbb{P}, \mathbb{Q}]$ as well. For this, let $p \in -P$; so $p \in P$. Then $f(p) \leq g(p)$ in $-\mathbb{Q}$. Thus $g(p) \leq f(p)$ in $\mathbb{Q}$. This for all $p \in P$ shows that $g \leq f$ in $[-\mathbb{P}, \mathbb{Q}]$.

Part (3) follows from parts (1) and (2).

For part (4), assume $x \leq y$. Then $f(x) \leq f(y)$, and so $g(f(x)) \leq g(f(y))$.

For part (5), note that by part (2), $g : -\mathbb{Q} \to \mathbb{R}$. So by part (4), $g \cdot f : \mathbb{P} \to \mathbb{R}$.

Continuing with part (6), note first that $-(\mathbb{P} \times \mathbb{Q})$ and $-\mathbb{P} \times -\mathbb{Q}$ have the same elements. To check that the orders are appropriately related, note that the following are equivalent: $(p, q) \leq (p', q')$ in $-(\mathbb{P} \times \mathbb{Q})$; $(p', q') \leq (p, q)$ in $(\mathbb{P} \times \mathbb{Q})$; $p' \leq p$ in $\mathbb{P}$ and $q' \leq q$ in $\mathbb{Q}$; $p \leq p'$ in $-\mathbb{P}$ and $q \leq q'$ in $-\mathbb{Q}$; $(p, q) \leq (p', q')$ in $-\mathbb{P} \times -\mathbb{Q}$.

For part (7), we verify that the identity map on P is an isomorphism from $\mathbb{P}$ to $-\mathbb{P}$. If $p \leq q$ in $\mathbb{P}$, then clearly $q \leq p$ in $-\mathbb{P}$. And if $q \leq p$ in $-\mathbb{P}$, then since $\mathbb{P}$ is flat, $p = q$. Thus $p \leq q$ in $\mathbb{P}$.

The last part comes from parts (3) and (8). ∎

Example 2.5. Part (3) in Proposition 2.4 is illustrated by considering $[\mathbb{2}, \mathbb{2}]$ and $[-\mathbb{2}, -\mathbb{2}]$. Let c be the constant function T, let d be the constant function F, and let i be the identity. Then $[\mathbb{2}, \mathbb{2}]$ is $d < i < c$, and $[-\mathbb{2}, -\mathbb{2}]$ is $c < i < d$.

Example 2.6. The material conditional $\to$ gives a monotone function from $-\mathbb{2} \times \mathbb{2}$ to $\mathbb{2}$. Since $[-\mathbb{2} \times \mathbb{2}, \mathbb{2}] \cong [-\mathbb{2}, [\mathbb{2}, \mathbb{2}]]$, this gives

$$\mathsf{if} \in [-\mathbb{2}, [\mathbb{2}, \mathbb{2}]].$$

This same function belongs to the opposite preorder, and so we could just as well write

$$\mathsf{if} \in -[-\mathbb{2}, [\mathbb{2}, \mathbb{2}]] \quad \cong \quad [\mathbb{2}, -[\mathbb{2}, \mathbb{2}]] \quad \cong \quad [\mathbb{2}, [-\mathbb{2}, -\mathbb{2}]].$$

Example 2.7. Let X be any set. We use letters like p and q to denote elements of $[\mathbb{X}, \mathbb{2}]$. Please keep in mind that the set of (monotone) functions from $\mathbb{X}$ to $\mathbb{2}$ is in one-to-one correspondence with the set of subsets of X (see Part (3) of Proposition 2.2). Define

$$\begin{array}{lcl} \mathsf{every} & \in & [-[\mathbb{X}, \mathbb{2}], [[\mathbb{X}, \mathbb{2}], \mathbb{2}]] \\ \mathsf{some} & \in & [[\mathbb{X}, \mathbb{2}], [[\mathbb{X}, \mathbb{2}], \mathbb{2}]] \\ \mathsf{no} & \in & [-[\mathbb{X}, \mathbb{2}], [-[\mathbb{X}, \mathbb{2}], \mathbb{2}]] \end{array}$$

in the standard way:

$$\begin{aligned}
\mathsf{every}(p)(q) &= \begin{cases} \mathsf{T} & \text{if } p \leq q \\ \mathsf{F} & \text{otherwise} \end{cases} \\
\mathsf{some}(p)(q) &= \neg\mathsf{every}(p)(\neg \cdot q) \\
\mathsf{no}(p)(q) &= \neg\mathsf{some}(p)(q)
\end{aligned}$$

It is routine to verify that these functions really belong to the sets mentioned above. And as in Example 2.6, each of these functions belongs to the opposite preorder as well, and we therefore have

$$\begin{array}{lcl}
\mathsf{every} & \in & [[\mathbb{X}, \mathbb{2}], [-[\mathbb{X}, \mathbb{2}], -\mathbb{2}]] \\
\mathsf{some} & \in & [-[\mathbb{X}, \mathbb{2}], [-[\mathbb{X}, \mathbb{2}], -\mathbb{2}]] \\
\mathsf{no} & \in & [[\mathbb{X}, \mathbb{2}], [[\mathbb{X}, \mathbb{2}], -\mathbb{2}]]
\end{array}$$

Summary The main point of this section is the fact mentioned in Proposition 2.4, part (3). In words, the monotone functions from $-\mathbb{P}$ to $-\mathbb{Q}$ are *exactly the same as* the monotone functions from $\mathbb{P}$ to $\mathbb{Q}$, but as preordered sets themselves, $[-\mathbb{P}, -\mathbb{Q}]$ and $[\mathbb{P}, \mathbb{Q}]$ are *opposite orders*. A simple illustration of the opposite orders was presented in Example 2.5. This fact, and also part (2) of Proposition 2.4, is important in the linguistic examples because it justifies why lexical items are typically assigned two categories. For example, if we interpret a common noun by an element f of a preorder of the form $[\mathbb{X}, \mathbb{2}]$, then the same f is an element of $-[\mathbb{X}, \mathbb{2}]$, so we might as well declare our noun to also have some typing which calls for an element of $-[\mathbb{X}, \mathbb{2}]$. And in general, if we type a lexical item w with a type (σ, τ) corresponding to an element of some preorder $[\mathbb{P}, \mathbb{Q}]$, then we might as well endow w with the type $(-\sigma, -\tau)$ with the very same interpretation function.

3. Higher-order Terms over Preorders

The centerpiece of this note is a presentation of the fully internalized type scheme. This section presents the details, aimed at readers familiar with categorical type logics (see Moortgat [7] for background). We should mention at the outset that what we are generalizing are the applicative (Ajdukiewicz-Bar Hillel) categorial grammars. However, everything we do extends to the more useful class of categorial grammars, as we shall see.

Fix a set $\mathcal{T}_0$ of *basic types*. Let $\mathcal{T}_1$ be the smallest superset of $\mathcal{T}_0$ closed in the following way:

If $\sigma, \tau \in \mathcal{T}_1$, then also $(\sigma, \tau) \in \mathcal{T}_1$.

If $\sigma \in \mathcal{T}_1$, then also $-\sigma \in \mathcal{T}_1$.

Let $\equiv$ be the smallest equivalence relation on $\mathcal{T}_1$ such that the following hold:

1. $-(-\sigma) \equiv \sigma$.
2. $-(\sigma, \tau) \equiv (-\sigma, -\tau)$.
3. If $\sigma \equiv \sigma'$, then also $-\sigma \equiv -\sigma'$.
4. If $\sigma \equiv \sigma'$ and $\tau \equiv \tau'$, then $(\sigma, \tau) \equiv (\sigma', \tau')$.

Definition $\mathcal{T} = \mathcal{T}_1/\equiv$. This is the set of *types over* $\mathcal{T}_0$.

The operations $\sigma \mapsto -\sigma$ and $\sigma, \tau \mapsto (\sigma, \tau)$ are well-defined on $\mathcal{T}$. We always use letters like σ and τ to denote elements of $\mathcal{T}$, as opposed to writing $[\sigma]$ and $[\tau]$. That is, we simply work with the elements of $\mathcal{T}_1$, but identify equivalent types.

Definition Let $\mathcal{T}_0$ be a set of basic types. A *typed language over* $\mathcal{T}_0$ is a collection of *typed variables* $v : \sigma$ and *typed constants* $c : \sigma$, where σ in each of these is an element of $\mathcal{T}$. We generally assume that the set of typed variables includes infinitely many of each type. But there might be no constants whatsoever. We use $\mathcal{L}$ to denote a typed language in this sense.

Let $\mathcal{L}$ be a typed language. We form *typed terms* $t : \sigma$ as follows:

1. If $v : \sigma$ (as a typed variable), then $v : \sigma$ (as a typed term).
2. If $c : \sigma$ (as a typed constant), then $c : \sigma$ (as a typed term).
3. If $t : (\sigma, \tau)$ and $u : \sigma$, then $t(u) : \tau$.

Frequently we do not display the types of our terms.

3.1. Semantics

For the semantics of our higher-order language $\mathcal{L}$ we use *models* $\mathcal{M}$ of the following form. $\mathcal{M}$ consists of an assignment of preorders $\sigma \mapsto \mathbb{P}_\sigma$ on $\mathcal{T}_0$,

together with some data which we shall mention shortly. Before this, extend the assignment $\sigma \mapsto \mathbb{P}_\sigma$ to $\mathcal{T}_1$ by

$$\begin{array}{lcl} \mathbb{P}_{(\sigma,\tau)} & = & [\mathbb{P}_\sigma, \mathbb{P}_\tau] \\ \mathbb{P}_{-\sigma} & = & -\mathbb{P}_\sigma \end{array}$$

An easy induction shows that if $\sigma \equiv \tau$, then $\mathbb{P}_\sigma = \mathbb{P}_\tau$. So we have $\mathbb{P}_\sigma$ for $\sigma \in \mathcal{T}$. We use P_σ to denote the set underlying the preorder $\mathbb{P}_\sigma$.

The rest of the structure of a model $\mathcal{M}$ consists of an assignment $[\![c]\!] \in P_\sigma$ for each constant $c : \sigma$, and also a *typed map* f; this is just a map which to a typed variable $v : \sigma$ gives some $f(v) \in P_\sigma$.

3.2. Ground terms and contexts

A *ground term* is a term with no free variables. Each ground term $t : \sigma$ has a *denotation* $[\![t]\!] \in P_\sigma$ defined in the obvious way:

$$\begin{array}{lcl} [\![c]\!] & = & \text{is given at the outset for constants } c : \sigma \\ [\![t(u)]\!] & = & [\![t]\!]([\![u]\!]) \end{array}$$

A *context* is a typed term with exactly one variable, x. (This variable may be of any type.) We write t for a context, or sometimes $t(x)$ to emphasize the variable. We'll be interested in contexts of the form $t(u)$. Note that if $t(u)$ is a context and if x appears in u, then t is a ground term; and vice-versa.

In the definition below, we remind you that subterms are not necessarily proper. That, is a variable x is a subterm of itself.

Definition Fix a model $\mathcal{M}$ for $\mathcal{L}$. Let $x : \rho$, and let $t : \sigma$ be a context. We associate to t a set function

$$f_t : P_\rho \to P_\sigma$$

in the following way:

1. If $t = x$, so that $\sigma = \rho$, then $f_x : P_\sigma \to P_\sigma$ is the identity.
2. If t is $u(v)$ with $u : (\tau, \sigma)$ and $v : \tau$, and if x is a subterm of u, then f_t is $app_{[\![v]\!]} \cdot f_u$. That is, $f_{t(u)}$ is
$$a \in P_\rho \mapsto f_u(a)([\![v]\!]) \ .$$
3. If t is $u(v)$ with $u : (\tau, \sigma)$ and $v : \tau$, and if x is a subterm of v, then f_t is $[\![u]\!] \cdot f_v$. That is, f_t is
$$a \in P_\rho \mapsto [\![u]\!](f_v(a)) \ .$$

The idea of f_t is that as a ranges over its interpretation space P_ρ, $f_t(a)$ would be the result of substituting various values of this space in for the variable, and then evaluating the result.

PROPOSITION 3.1. *For all contexts $t(x)$ with $t : \sigma$ and $x : \rho$, and all ground terms $s : \rho$,*

$$f_t([\![s]\!]) \quad = \quad [\![t(x \leftarrow s)]\!],$$

where $t(x \leftarrow u)$ is the result of substituting s for x in t.

PROOF. By induction on t. If $t = x$, the result is trivial. Suppose that $t = u(v)$ with x appearing in v. Then $t(x \leftarrow s)$ is $u(x \leftarrow s)(v)$, and

$$f_t([\![s]\!]) = f_u([\![s]\!])([\![v]\!]) = [\![u(x \leftarrow u)]\!]([\![v]\!]) = [\![u(x \leftarrow u)(v)]\!] = [\![t(x \leftarrow s)]\!]$$

Finally, if $t = u(v)$ with x appearing in v, then $t(x \leftarrow s)$ is $u(v(x \leftarrow s))$, and

$$f_t([\![s]\!]) \quad = \quad [\![u]\!](f_v([\![s]\!])) \quad = \quad [\![u(v(x \leftarrow s))]\!] \quad = \quad [\![t(x \leftarrow s)]\!]$$

In both calculations, we used the induction hypothesis. ∎

Notice that we defined f_t as a set function and wrote $f_t : P_\rho \to P_\sigma$ instead of $f_t : \mathbb{P}_\rho \to \mathbb{P}_\sigma$. The reason why we did this is that it is not immediately clear that f_t is monotone. This is the content of the following result.

LEMMA 3.2 (Context Lemma). *Let t be a context, where $t : \sigma$ and $x : \rho$. Then f_t is element of $[\mathbb{P}_\rho, \mathbb{P}_\sigma]$.*

PROOF. By induction on t. In the case that t is a type variable $x : \sigma$, then f_x is the identity on $\mathbb{P}_\sigma$, and indeed the identity is a monotone function on any preorder.

Next, assume that x occurs in u. By induction hypothesis, f_u belongs to $[\mathbb{P}_\rho, \mathbb{P}_{(\tau,\sigma)}] = [\mathbb{P}_\rho, [\mathbb{P}_\tau, \mathbb{P}_\sigma]]$. The function $app_{[\![u]\!]}$ belongs to $[[\mathbb{P}_\tau, \mathbb{P}_\sigma], \mathbb{P}_\sigma]$ by Proposition 2.2, part (1), and so f_t belongs to $[\mathbb{P}_\rho, \mathbb{P}_\sigma]$ by Proposition 2.4, part (4).

Finally, assume that x occurs in v. By induction hypothesis, $f_v : \mathbb{P}_\rho \to \mathbb{P}_\tau$. And $[\![u]\!] \in \mathbb{P}_{(\tau,\sigma)}$, so that $[\![u]\!] : \mathbb{P}_\tau \to \mathbb{P}_\sigma$. By Proposition 2.4, part (4), $[\![u]\!] \cdot f_v$ is an element of $[\mathbb{P}_\rho \to \mathbb{P}_\sigma]$. ∎

This Context Lemma implies that the idea of internalized polarity markers works. We'll see some examples in the next section which clarify this point. Notice also that it is a very general result: the only facts about preorders were the very general results in Section 2.

The Context Lemma also allows one to generalize our work from the applicative (AB) grammars to the setting of CG using the Lambek Calculus. In detail, one generalizes the notion of a context to one which allows more than one free variable but requires that all variables which occur free have only one free occurrence. Suppose that $x_1 : \rho_1, \ldots, x_n : \rho_n$ are the free variables in such a generalized context $t(x_1, \ldots, x_n) : \sigma$. Then t defines a set function a set function $f_t : \prod_i P_{\rho_i} \to P_\sigma$. A generalization of the Context Lemma then shows that f_t is monotone as a function from the product preorder $\prod_i \mathbb{P}_{\rho_i}$. So functions given by lambda abstraction on each of the variables are also monotone, and this amounts to the soundness of the introduction rules of the Lambek Calculus in the internalized setting.

This point an advantage of the internalized system in comparison to the monotonicity calculus: in the latter approach, the results of polarity determination become side conditions on the introduction rules for the lambda operator.

3.3. Logic

Figure 2 several sound logical principles. These are implicit in Fyodorov, Winter, and Francez [4]; see also Zamansky, Francez, and Winter [14] for a more developed proposal. The rules are the reflexive and transitive properties of the preorder, the fact that all function types are interpreted by sets of monotone functions, and the pointwise definition of the order on function sets. The rules in Figure 2 define a logical system whose assertions order statements such as $u : \sigma \leq v : \sigma$; then the statements such as $u : \sigma$ become *side conditions* on the rules. (For simplicity, we are only dealing with ground terms u and v, but it is not hard to generalize the treatment to allow variables.) The logic thus defines a relation $\Gamma \vdash \varphi$ on order statements. Given a model $\mathcal{M}$ and an order statement ψ of the form $u : \sigma \leq v : \sigma$, we say that $\mathcal{M}$ *satisfies* ψ iff $[\![u]\!] \leq [\![v]\!]$ in $\mathbb{P}_\sigma$. The soundness of the logic is the statement that every model $\mathcal{M}$ satisfying all of the sentences in Γ also satisfies ϕ.

The Context Lemma then gives another sound principle:

$$\frac{x : \sigma \text{ in the context } t \quad u : \sigma \leq v : \sigma}{t(u) : \tau \leq t(v) : \tau} \tag{1}$$

However, all instances of (1) are already provable in the logic as we have defined it. This is an easy inductive argument on the context t.

$$\frac{}{t:\sigma \leq t:\sigma} \qquad \frac{t:\sigma \leq u:\sigma \quad u:\sigma \leq v:\sigma}{t:\sigma \leq v:\sigma}$$

$$\frac{u:\sigma \leq v:\sigma \quad t:(\sigma,\tau)}{t(u):\tau \leq t(v):\tau} \qquad \frac{u:(\sigma,\tau) \leq v:(\sigma,\tau) \quad t:\sigma}{u(t):\tau \leq v(t):\tau}$$

Figure 2. Monotonicity Principles

4. Examples and Discussion

We present a small example to illustrate the ideas. It is a version of the language $\mathcal{R}^*$ from Pratt-Hartmann and Moss [9]. We also take the opportunity to discuss internalized marking in a general way. We have a few comments on the challenges of building a logic based on internalized markings. And we also have a discussion of the word *any* and how it might be treated.

First, we describe a language $\mathfrak{L}$ corresponding to this vocabulary. Let's take our set $\mathcal{T}_0$ of basic types to be $\{t, pr\}$. (These stand for *truth value* and *property*. In more traditional presentations, the type pr might be (e,t), where e is a type of *entities*.)

Here are the constants of the language $\mathfrak{L}$ and their types:

1. We have typed constants

$$\begin{array}{lcllcl} \mathsf{every}^+ & : & (-pr,(pr,t)) & \mathsf{every}^- & : & (pr,(-pr,-t)) \\ \mathsf{some}^+ & : & (pr,(pr,t)) & \mathsf{some}^- & : & (-pr,(-pr,-t)) \\ \mathsf{no}^+ & : & (-pr,(-pr,t)) & \mathsf{no}^- & : & (pr,(pr,-t)) \\ \mathsf{any}^+ & : & (-pr,(pr,t)) & \mathsf{any}^- & : & (-pr,(-pr,-t)) \end{array}$$

2. We fix a set of *unary atoms* corresponding to some plural nouns and lexical verb phrases in English. For definiteness, we take cat, dog, animal, runs, sleeps. Each unary atom p gives two typed constants: $\mathsf{p}^+ : pr$ and $\mathsf{p}^- : -pr$.
3. We also fix a set of *binary atoms* corresponding to some transitive verbs in English. To be definite, we take chase, see. Every binary atom r gives four type constants:

$$\begin{array}{lcllcl} \mathsf{r}_1^+ & : & ((pr,t),pr) & \mathsf{r}_2^+ & : & ((-pr,t),pr) \\ \mathsf{r}_1^- & : & ((-pr,-t),-pr) & \mathsf{r}_2^- & : & ((pr,-t),-pr) \end{array}$$

This completes the definition of our typed language $\mathfrak{L}$.

We might mention that the superscripts + and − on the constants $c : \sigma$ of $\mathfrak{L}$ are exactly the polarities $\mathrm{pol}(\sigma)$ of the associated types. (We shall define the polarity function in the next section.) These notations + and − are mnemonic; we could do without them.

As always with categorial grammars, a *lexicon* is a set of pairs consisting of words in a natural language together with terms. We have been writing the words in the target language in *italics*, and then terms for them are written in sans serif. It is very important that the lexicon allows a given word to appear with many terms. As we have seen, we need *every* to appear with every^+ and every^-, for example. We still are only concerned with the syntax at this point, and the semantics will enter once we have seen some examples.

Examples of typed terms and contexts Here are a few examples of typed terms along with their derivations. First, the bottommost derivation in Figure 1 is a derivation of the term corresponding to the English sentence *No dog chases every cat*. One should compare this treatment with what we saw of the same sentence in the Introduction, indicated in the top two derivations in the figure. The internalized setting does not have steps of Monotonicity Marking and Polarity Determination. It only involves a derivation in the applicative CG at hand. Please also keep in mind that the superscripts + and − above are from the lexicon, not from any additional process, and that the lexicon could have been formulated without those superscripts. They are in fact the polarities of the attendant categories.

We similarly have the following terms:

$$\begin{array}{l}\mathsf{some}^+(\mathsf{dog}^+)(\mathsf{chase}_1^+(\mathsf{every}^+(\mathsf{cat}^-))) : t\\ \mathsf{some}^+(\mathsf{dog}^+)(\mathsf{chase}_2^+(\mathsf{no}^+(\mathsf{cat}^-))) : t\\ \mathsf{no}^+(\mathsf{dog}^-)(\mathsf{chase}_2^-(\mathsf{no}^+(\mathsf{cat}^+))) : t\end{array}$$

The point here is that all four different typings of the transitive verbs are needed to represent sentences of English. I do not believe that this point has been noticed about the internalized scheme before. It raises an issue that we shall take up in the next subsection: how would a grammar writer come up with all of the needed categories?

Here is an example of a context:

$$u(x) \quad \equiv \quad \mathsf{no}^+(x : -pr)(\mathsf{chase}_1^-(\mathsf{every}^-(\mathsf{cat}^+))) : t.$$

In any model, this context gives a function f_u from interpretations of type $-pr$ to those of type t. The Context Lemma would tell us that this function

is a monotone function:

$$f_u : \mathbb{P}_{-pr} \to \mathbb{P}_t$$

By Proposition 2.4, part (3), we can also write

$$f_u : \mathbb{P}_{pr} \to \mathbb{P}_{-t}$$

In other words, the interpretation of the function is an *antitone* function from property interpretations to truth value interpretations.

Any The internalized approach enables a treatment of *any* that has it *any* mean the same thing as *every* when it has positive polarity, and the same thing as *some* when it has negative polarity. For example, here is a sentence intended to mean *everything which sees any cat runs*:

$$\mathsf{every}^+(\mathsf{see}_2^-(\mathsf{any}^-(\mathsf{cat}^-))(\mathsf{runs}^+) : t$$

The natural reading is for *any* to have an existential reading. Another context for existential readings of *any* is in the antecedent of a conditional. In the other direction, consider

$$\mathsf{any}^+(\mathsf{cat}^-)(\mathsf{see}_1^-(\mathsf{any}^+(\mathsf{dog}^-))) : t.$$

The natural reading of both occurrences is universal.

4.1. Standard models

Up until now, we have only given one example of a typed language $\mathcal{L}$. Now we describe a family of models for this language. The family is based on sets with interpretations of the unary and binary atoms. To make this precise, let us call a *pre-model* a structure $\mathcal{M}_0$ consisting of a set M together with subsets $[\![\mathsf{p}]\!] \subseteq M$ for unary atoms p and relations $[\![\mathsf{r}]\!] \subseteq M \times M$ for binary atoms r.

Every pre-model $\mathcal{M}_0$ now gives a bona fide model $\mathcal{M}$ of $\mathcal{L}$ in the following way. The underlying universe M gives a flat preorder $\mathbb{M}$. We take $\mathbb{P}_{pr} = [\mathbb{M}, \mathbb{2}] \cong \mathcal{P}(M)$. We also take $\mathbb{P}_t = \mathbb{2}$.

We interpret the typed constants $\mathsf{p}^+ : pr$ corresponding to unary atoms by

$$[\![\mathsf{p}^+]\!](m) = \mathsf{T} \quad \text{iff} \quad m \in [\![\mathsf{p}]\!].$$

(On the right we use the interpretation of p in the model $\mathcal{M}$.) Usually we write p^+ instead of $[\![\mathsf{p}]\!]$.

The constants $\mathsf{p}^- : -pr$ are interpreted by the same functions.

The interpretations of every^+, some^+, and no^+ are given in Example 2.7 (taking the set X to be the universe M of $\mathcal{M}$). And the interpretations of every^-, some^-, and no^- are the same functions.

We interpret any^+ the same way as every^+, and any^- the same way as some^+. It is important at this point to check that this interpretation respects the types. That is,

$$\begin{array}{rcl} \mathsf{any}^+ & = & \mathsf{every}^+ \in [-[\mathbb{M},\mathbb{2}],-[[\mathbb{M},\mathbb{2}],\mathbb{2}]] \quad = \quad \mathbb{P}_{(-pr,(pr,t))} \\ \mathsf{any}^- & = & \mathsf{some}^- \in [[\mathbb{M},\mathbb{2}],[[\mathbb{M},\mathbb{2}],-\mathbb{2}]] \quad = \quad \mathbb{P}_{(pr,(pr,-t))} \end{array}$$

Some calculations relevant to these typings were given in Example 2.7.

Recall that the binary atom r gives four typed constants r_1^+, r_1^-, r_2^+, and r_2^-. These are all interpreted in model in the same way, by

$$\mathsf{r}_*(q)(m) \quad = \quad q(\{m' \in M : [\![r]\!](m,m')\})$$

It is clear that for all $q \in \mathbb{P}_{(pr,t)} = [[\mathbb{M},\mathbb{2}],\mathbb{2}]$, $\mathsf{r}_*(q)$ is a function from M to $\{\mathsf{T},\mathsf{F}\}$. The monotonicity of this function is trivial, since $\mathbb{M}$ is flat.

We do need to check that all of the interpretations of the constants are correctly typed. This is important, and it raises a general issue.

Issue 1: What are the sources of the multiple typings of the lexical items? As this short paper draws to a close, we wish to identify some issues with the internalized scheme of polarity marking. Here is the first: The scheme requires many entries for each lexical item, usually with the same semantics in every model. These amount to *multiple typings of the same function*. What is the source of these multiple typings, and can they be automatically inferred?

The principal source for multiple typings is the fact noted in Proposition 2.4, part (3): $[-\mathbb{P},-\mathbb{Q}] = -[\mathbb{P},\mathbb{Q}]$. That is, we have two (opposite) order relations on the same set, and so it makes sense that we should type an element of that set in two ways. This accounts for the multiple typings of nouns and determiners in our example, but not for transitive verbs. For them, and for all categories in more complicated languages, we suspect that there is no simple answer. Perhaps the best one could hope for would be a formal system in which to derive the various typings. In the same way that one could propose logical systems which process inferences in parallel with syntactic derivations, one could hope to do the same thing for order statements. Here is our vision of a what a derivation might look like in such

a system. Let us write $\phi(m, \mathsf{r})$ for $\{m' \in M : [\![r]\!](m, m')\}$. A justification for the typing $\mathsf{r}_2^- : [[[\mathbb{M}, \mathbb{2}], -\mathbb{2}], [\mathbb{M}, -\mathbb{2}]]$ could be the derivation below:

$$\dfrac{\dfrac{\dfrac{q \leq q' \text{ in } [[\mathbb{M}, \mathbb{2}], -\mathbb{2}] \quad \phi(m, \mathsf{r}) \in [\mathbb{M}, \mathbb{2}]}{\mathsf{r}_2^-(q) = q(\phi(m, \mathsf{r})) \leq q'(\phi(m, \mathsf{r})) = \mathsf{r}_2^-(q') \text{ in } -\mathbb{2}}}{\mathsf{r}_2^-(q) \leq \mathsf{r}_2^-(q') \text{ in } [\mathbb{M}, -\mathbb{2}]}}{\mathsf{r}_2^- : [[[\mathbb{M}, \mathbb{2}], -\mathbb{2}], [\mathbb{M}, -\mathbb{2}]]}$$

The challenge would be to propose a system which is small enough to be decidable, and yet big enough to allow us to use the notation $\phi(m, \mathsf{r})$ as we have done it.

Here is a related point. As Dowty [3] notes, the parsing problem for internalized grammars is no harder than that of CG in the first place. In fact, the work for internalized grammars is a *special case*, since they are CGs with nothing extra. However, this could be a misleading comparison: if one is given a CG $\mathcal{G}$ and some additional information (for example, the intended meanings of the constants corresponding to determiners), one would need to convert this to an internalized grammar $\mathcal{G}^*$. It may be that the size of the lexicon of $\mathcal{G}^*$ is exponential in the size of the lexicon of $\mathcal{G}$. The fact that in our small grammar we need four typings for the verbs makes one worry. And in the worst case, the internalized scheme would be less attractive for computational purposes.

Issue 2: what is the relation of polarity and monotonicity? It is "well-known" in this area that negative polarity items are those which must occur, or usually occur in "downward monotone positions" in a given sentence. But without saying what the orders are on the semantic spaces, this point could be confusing. For example, suppose one has a reason to assume that the order on truth values is $\mathsf{T} < \mathsf{F}$ rather than $\mathsf{F} < \mathsf{T}$. In this case, the notions of upward monotone positions and downward monotone ones would switch. More seriously, given a complicated function type $(\sigma, -\tau)$, it is not so clear whether this should be recast as $-(-\sigma, \tau)$. In this case, it would not be so clear what the upward and downward monotone positions are in a sentence.

This matter might be clarified by looking at *no* vs. *at most one* in a sentence such as *No dog chases every cat.* As we have seen, our treatment suggests that the occurrence of *no* is in a positive position, and so we expect it to be monotone increasing. We would like to say that $\mathsf{no} \leq \mathsf{at\ most\ one}$, since this is the verdict from generalized quantifier theory. Indeed, the interpretations functions no and $\mathsf{at\ most\ one}$ are taken to be functions of type

$((M,2),((M,2),2))$, and then the order on this space is taken to be ultimately determined by the last "2". However, we do not have a single function no but rather two functions, no^+ and no^-. These are *not quite* restrictions of no and at most one. For that matter, why exactly do we say that no^+ is monotone increasing? It's type was $(-pr,(-pr,t))$, after all. What seems to be going on here is that in a curried form, we have $(-pr \times -pr, t)$. When curried like this, no^+ and no^- *are* restrictions of no and at most one. Moreover, the order on on a product space is determined by the codomain order, forgetting the domain order. Also, the "meaning space" in a model for types pr and $-pr$ are the same. Taken together, this means that no^+ indeed corresponds to a monotone function and no^- to an antitone function.

Here is a general definition of the *polarity of a type*. Define $\mathrm{pol}(\sigma) : \mathcal{T}_1 \to \{+,-\}$ as follows:

1. If $\sigma \in \mathcal{T}_0$, then $\mathrm{pol}(\sigma) = +$.
2. $\mathrm{pol}((\sigma,\tau)) = \mathrm{pol}(\tau)$.
3. $\mathrm{pol}(-\sigma) = -\mathrm{pol}(\sigma)$.

It is easy to check that if $\sigma \equiv \tau$, then $\mathrm{pol}(\sigma) = \mathrm{pol}(\tau)$. (It is enough to show that $\mathrm{pol}(-(-\sigma)) = \mathrm{pol}(\sigma)$, and also that $\mathrm{pol}(-(\sigma,\tau)) = \mathrm{pol}((-\sigma,-\tau))$.) This allows us define pol on the quotient set, $\mathcal{T}_1/\equiv$, our set $\mathcal{T}$ of types.

The definition of the pol function gives an algorithm to calculate $\mathrm{pol}(\sigma)$. Another way to do this would be to: (1) put σ in "negation normal form", by driving the - signs through function spaces so that there are no function types inside the scope of any $-$ sign; (2) remove an even number of $-$ signs from basic types, so that the remaining basic types have either zero or one $-$ sign in front; (3), finally, starting at the top, proceed "to the right" through the term until one reaches either a basic type or its negation.

For example, every^- is of type $(pr,(-pr,-t))$, and the polarity of its occurrence in our example sentence *No dog chases every cat* (or in any term derived in our grammar), is $-$.

Let u be a subterm occurrence in the term t. Then there is a unique context $t'(x)$ such that $t = t'[u \leftarrow x]$. Let σ be the type of x in t'. We say that u has *positive polarity* if $\mathrm{pol}(x) = +$, and that u has *negative polarity* if $\mathrm{pol}(x) = -$.

We naturally would like to say that if an occurrence u has positive polarity in t, then that occurrence is in a monotone increasing position, and if the occurrence has negative polarity, then it is in a monotone decreasing position.

Issue 3: how can we build a logic to reason about classes of standard models? Our last issue concerns the contribution of the types in an internalized CG to a natural logic. The matter is illustrated by a discussion concerning the example language $\mathcal{L}$ presented at the beginning of Section 4. Suppose we are given *entailment* relations concerning the unary and binary atoms of $\mathcal{L}$. For unary atoms, these take the form $\mathsf{p} \Rightarrow \mathsf{q}$, where p and q are unary atoms. For binary atoms, entailment relations look similar: $\mathsf{r} \Rightarrow \mathsf{s}$. (Incidentally, one is also interested in *exclusion relations*: see MacCartney and Manning [6] for applications, and Icard [5] for theory behind this.)

These entailment relations correspond to semantic restrictions on standard models. The first corresponds to the requirement that $[\![\mathsf{p}]\!] \subseteq [\![\mathsf{q}]\!]$, and the second to the parallel requirement that $[\![\mathsf{r}]\!] \subseteq [\![\mathsf{s}]\!]$. Working with entailment relations in this way means restricting attention to standard models which conform to the restrictions. It is natural to then look for additional axioms and/or rules of inference in the logical system that would give a sound and complete logic for these models. The entailment relations correspond to *axioms* of the sentential kind, and also on the orders. The sentential axioms can be stated only for the unary atoms, and they would be sentences like $(\mathsf{every}^+(\mathsf{p}^-))\mathsf{q}^-$. For the binary atoms, the statements are more involved. To state one on an informal level, suppose one wants to model a fact such as *kissing* (r) entails *touching* (s) using an entailment fact of the form $\mathsf{r} \Rightarrow \mathsf{s}$. Then a logical system should contain an axiom corresponding to a sentence such as *Everyone who kisses some baby touches some baby.* Turning to axioms on the orders, our entailment relations give us the following:

$\mathsf{p}^+ \leq \mathsf{q}^+$ in pr	$\mathsf{r}_1^- \leq \mathsf{s}_1^+$ in $((-pr, -t), -pr)$
$\mathsf{q}^- \leq \mathsf{p}^-$ in $-pr$	$\mathsf{r}_2^+ \leq \mathsf{s}_2^+$ in $((-pr, t), pr)$
$\mathsf{r}_1^+ \leq \mathsf{s}_1^+$ in $((pr, t), pr)$	$\mathsf{r}_2^- \leq \mathsf{s}_2^-$ in $((pr, -t), -pr)$

However, even adopting both kinds of axioms on top of the logical principles we saw in Section 3.3 would not result in a complete logical system, so it would fall short of what we want in a natural logic. One source of problems was noted in Zamansky, Francez, and Winter [14]: the system would not have any way to derive principles corresponding to de Morgan's laws. Another problem for us would be even more basic. The resulting logic does not appear to be strong enough to derive the sound sentences of the form *every X is an X*. To be sure, there are unsound sentences of this form in the logic, due to the typing. For example, the way we are doing things, *every dog who chases any cat chases any cat* is not a logical validity at all, since the first *any* is to be interpreted existentially and the second universally. (I first heard an example of this type in a lecture of Barbara Partee.) But without *any*, we

would like to derive, for example

$$\mathsf{every}^{+}(\mathsf{see}_1^{-}(\mathsf{every}^{-}(\mathsf{cat}^{+})))(\mathsf{see}_1^{+}(\mathsf{every}^{+}(\mathsf{cat}^{-})))$$

At this point, I can merely speculate and offer a suggestion. My feeling is that one should look to logics with *individual variables*, and then the properties of the individual words become rules of inference in the logic. Specifically, it should be the case that if x is an individual or NP-level variable, then one should be able to infer $\mathsf{cat}^{+}(x)$ from $\mathsf{cat}^{-}(x)$, and vice-versa. Logics of this type have been proposed in the literature, but they do not build on the framework of categorial grammar and therefore have nothing to say about polarity and monotonicity. Integrating the work there with the proposal in this paper is the most important open issue in work on this topic.

Conclusion This paper explores what happens to CG if we adopt the idea that interpretation should take place in *ordered* domains, especially if we add to the type formation rules a construction for the opposite order. Doing this gives a Context Lemma in a straightforward way, and the Context Lemma is tantamount to the soundness of the internalized polarity marking scheme proposed by Dowty [3] as a modification of the earlier ideas of van Benthem [11], and Sánchez Valencia [10]. The types in the internalized scheme are more plentiful than in ordinary CG, and this is a source of issues as well as possibilities. The issues concern how the new typings are to be generated, and how they are to interact with each other in proof-theoretic settings. There certainly is more to do on this matter, and this paper has only provided the groundwork. With some additional work, one could easily imagine that the internalized scheme will be an important contribution to the research agenda of natural logic.

Acknowledgements. My thanks to Thomas Icard and to Yoad Winter for their comments on an earlier draft of this paper.

References

[1] Raffaella Bernardi. *Reasoning with Polarity in Categorial Type Logic.* PhD thesis, University of Utrecht, 2002.

[2] Christos Christodoulopoulos. Creating a natural logic inference system with combinatory categorial grammar. Master's thesis, University of Edinburgh, 2008.

[3] David Dowty. The role of negative polarity and concord marking in natural language reasoning. In *Proceedings of SALT IV*, page 114144. Cornell University, Ithaca, NY, 1994.

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

[5] Thomas Icard. Exclusion and containment in natural language. ms., Stanford University, 2010.

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

[7] Michael Moortgat. Categorial type logics. In *Handbook of Logic and Language*, pages 93–178. MIT Press, Cambridge, MA, 1997.

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

[9] Ian Pratt-Hartmann and Lawrence S. Moss. Logics for the relational syllogistic. *Review of Symbolic Logic*, 2(4):647–683, 2009.

[10] Víctor M. Sánchez Valencia. *Studies on natural logic and categorical grammar*. PhD thesis, University of Amsterdam, 1991.

[11] Johan van Benthem. *Essays in Logical Semantics*, volume 29 of *Studies in Linguistics and Philosophy*. D. Reidel Publishing Co., Dordrecht, 1986.

[12] Johan van Benthem. *Language in Action*, volume 130 of *Studies in Logic and the Foundations of Mathematics*. North Holland, Amsterdam, 1991.

[13] Johan van Benthem. A brief history of natural logic. In M. Chakraborty, B. Löwe, M. Nath Mitra, and S. Sarukkai, editors, *Logic, Navya-Nyaya and Applications, Homage to Bimal Krishna Matilal*. College Publications, London, 2008.

[14] Anna Zamansky, Nissim Francez, and Yoad Winter. A 'natural logic' inference system using the Lambek calculus. *J. Log. Lang. Inf.*, 15(3):273–295, 2006.