A Complete Calculus of Monotone and Antitone Higher-Order Functions

Thomas F. Icard III[1] and Lawrence S. Moss[2*]

[1] Stanford University Philosophy Department, Stanford CA. icard@stanford.edu
[2] Indiana University Mathematics Department, Bloomington, IN. lsm@cs.indiana.edu

It has long been recognized that many expressions in natural language can be thought of as *monotone* and *antitone* operators. If we take two preordered domains $(P_1, \leq_1)$ and $(P_2, \leq_2)$, then a function $f : P_1 \to P_2$ is monotone if for all $a, b \in P_1$, if $a \leq_1 b$ then $f(a) \leq_2 f(b)$. A function is antitone if $a \leq_1 b$ implies $f(b) \leq_2 f(a)$. A natural language example of a monotone operator is Some [$^+$] is nocturnal, since replacing X in [$^+$] with any $Y \supseteq X$ preserves truth: Some [porcupine] is nocturnal implies Some [rodent] is nocturnal. By contrast, Every [$^-$] is nocturnal is antitone in exactly the analogous sense: Every [rodent] is nocturnal implies Every [porcupine] is nocturnal, but not vice versa.

Monotonicity is a pervasive feature of natural language (NL), and has been linked to many fundamental aspects of processing, reasoning, and even grammar. In the late 1980s van Benthem [5, 6] and Sánchez-Valencia [4] defined proof systems for reasoning about entailment in NL 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/antitonicity information, and use this in a proof system. For instance, an occurrence of every in an expression might be assigned a *marked type* $p \xrightarrow{-} (p \xrightarrow{+} t)$, capturing the fact that it is antitone in its first argument, monotone in its second. Though soundness of the main two proof rules (our (Mono) and (Anti) below in Def. 9) can be demonstrated [5, 6, 4], much of this work is informal, and so much is implicit. In particular, there are no completeness results, and even some of the delicate formal details of the type system, the language, and its interpretation are less developed.

Work on this topic has been revived by computational linguists; see [2, 3] for example. Although our ultimate aim is the application to NL semantics and to computational linguistics, we hold that the issues of monotonicty and antitonicity are of mathematical interest to the TACL community.

In this paper, we provide mathematical underpinnings of the Monotonicty Calculus. We devise a type system that includes marked types at any level of the functional hierarchy, and define languages of typed terms made up of constants by function applications (thus, no variables). Our main interest is a proof calculus for deriving inequality statements between typed terms. We prove soundness and completeness of the calculus with respect to the natural class of models, hierarchies of monotone and antitone functions over base preorders.

Types and Structures

Definition 1. Working over some set $\mathcal{B}$ of basic types, the full set of types $\mathcal{T}$ is defined as the smallest containing $\mathcal{B}$, such that whenever $\sigma, \tau \in \mathcal{T}$, so is $\sigma \xrightarrow{m} \tau$, for $m \in \mathcal{M} = \{+, -, \cdot\}$.

Expressions of type $\sigma \xrightarrow{+} \tau$ will denote monotone functions, those of type $\sigma \xrightarrow{-} \tau$ antitone functions, and those of type $\sigma \xrightarrow{\cdot} \tau$ arbitrary functions. We therefore have a natural preorder

*This work was partially supported by a grant from the Simons Foundation (#245591 to Lawrence Moss).

on $\mathcal{M}$, whereby $m \sqsubseteq m'$ iff $m = m'$ or $m' = \cdot$. This ordering can be used to define a natural preorder on types. Intuitively $\sigma \preceq \tau$ will mean, 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."

Definition 2 ($\preceq$ 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 1. As mentioned above, we shall use a running example from natural language semantics because it is the main motivation for our work. For expository purposes in this abstract, our example is highly oversimplified. We use base types p ("property") and t ("truth value"). 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. 2, $p \overset{-}{\to} (p \overset{+}{\to} t) \preceq p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$.

Definition 3. 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. $\uparrow$ 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 markings $m_1, m_2 \in \mathcal{M}$, and
$$(\sigma \overset{m_1}{\to} \tau_1) \vee (\sigma \overset{m_2}{\to} \tau_2) \quad = \quad \sigma \xrightarrow{m_1 \vee m_2} (\tau_1 \vee \tau_2).$$

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 that are "tamed" by Definition 3.

Example 2. Returning to Ex. 1, we have $p \overset{-}{\to} (p \overset{+}{\to} t) \ \uparrow \ p \overset{+}{\to} (p \overset{+}{\to} t) \ \uparrow \ p \overset{-}{\to} (p \overset{-}{\to} t)$, and so on. The least upper bound for this $\uparrow$-equivalence class is $p \overset{\cdot}{\to} (p \overset{\cdot}{\to} t)$.

The full version of this paper interprets the language on a class of *applicative structures* as the basic models. (This is a parallel to the use of *general models* in dealing with second-order logic, and also in Friedman's use of applicative structures in [1].) For reasons of space, we focus here on the standard structures. The completeness result for standard structures is stronger because the class of models is smaller.

Definition 4 (Structures)**.** A *standard structure* is a system $\mathcal{S} = \{\mathbb{D}_\tau\}_{\tau \in \mathcal{T}}$ of preordered *type domains* $\mathbb{D}_\tau = (D_\tau, \leq_\tau)$, where $\mathbb{D}_\beta$ is given antecedently for basic types $\beta \in \mathcal{B}$, and for functional types $\sigma \overset{m}{\to} \tau$: $D_{\sigma \overset{+}{\to} \tau}$ is the set of all monotone functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$, $D_{\sigma \overset{-}{\to} \tau}$ the set of all antitone functions, and $D_{\sigma \overset{\cdot}{\to} \tau}$ the set of all functions. In all cases, the resulting set $D_{\sigma \overset{m}{\to} \tau}$ is endowed with the pointwise order to obtain a preorder $\mathbb{D}_{\sigma \overset{m}{\to} \tau}$. In this abstract, we assume that each $\mathbb{D}_\beta$ has an additional *weak completeness* property: every two points have an upper bound and a lower bound.

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 make ordering statements between expressions of $\uparrow$-related types in the formal language.

Example 3. Usually one takes $\mathbb{D}_p$ to be an arbitrary boolean algebra and $\mathbb{D}_t$ to be the two-element BA.

Language and Interpretation

Definition 5 (Unlabeled Typed Terms)**.** We begin the syntax with a set Con of *constants* together with a function $\mathsf{type} : \mathsf{Con} \to \mathcal{T}_{\mathcal{M}}$. 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.

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.

Example 4. 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) $\mathsf{every} : p \overset{-}{\to} (p \overset{+}{\to} t)$, $\mathsf{some} : p \overset{+}{\to} (p \overset{+}{\to} t)$, $\mathsf{no} : p \overset{-}{\to} (p \overset{-}{\to} t)$, and $\mathsf{most} : p \overset{\cdot}{\to} (p \overset{+}{\to} t)$. Transitive verbs like see could have type $(p \overset{\cdot}{\to} t) \overset{+}{\to} p$. Then sentences of the form det+noun+verb+det+noun would correspond to terms of type t. (See Example 5.)

Definition 6 (Denotation)**.** For each term $t : \tau$, we define $[\![t]\!]$ by induction on t.

1. For the base case of a constant $c : \tau$, $[\![c]\!] \in D_\tau$ is stipulated, respecting monotonicity markings.
2. If $t : \sigma \overset{m}{\to} \tau$ and $u : \sigma'$ with $\sigma' \preceq \sigma$, then $[\![t(u)]\!] = [\![t]\!]\big([\![u]\!]\big)$.

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. For instance, if $t : \sigma \overset{-}{\to} \tau$ and $u : \rho \overset{-}{\to} \sigma$, then in $t(u(v)) : \tau$, subterm $v : \rho$ is in a monotone position. There is a simple algebra of markings $(\mathcal{M}, \circ)$, whereby $\circ$ is associative and commutative, and: $m \circ + = m$, $m \circ \cdot = \cdot$, and $- \circ - = +$. (This tiny algebra is the basis of much work on natural logic [7, 4, 8, 9].) We use this algebra to define terms labeled with their monotonicity information.

Definition 7 (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 $s : \tau \overset{m}{\to} \sigma$, 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 $\overset{m}{\to}$, and l is the label of the occurrence inside v.

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 $[\![u]\!] \leq_{\hat{\rho}} [\![v]\!]$:

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

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.

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 8 (Satisfaction)**.** Where $s : \sigma$ and $t : \tau$, and if $\sigma \uparrow \tau$, we write $\mathcal{S} \models s \leq t$ if $[\![s]\!] \leq_{\hat{\sigma}} [\![t]\!]$. 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 9 (Monotonicity Calculus)**.** The Monotonicity Calculus is given by the following rules:

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

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

$$(\textsc{Point})\ \frac{s \leq t}{s(u) \leq t(u)} \qquad (\text{WC1})\ \frac{f^\downarrow \leq g^\uparrow}{f^\downarrow(a) \leq g^\uparrow(b)} \qquad (\text{WC2})\ \frac{f^\uparrow \leq g^\downarrow}{f^\uparrow(a) \leq g^\downarrow(b)}$$

If Γ is any 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 the inequalities in Γ using the rules above.

Example 5. Let Γ contain $\mathsf{every} : p \xrightarrow{-} (p \xrightarrow{+} t) \leq \mathsf{most} : p \xrightarrow{\cdot} (p \xrightarrow{+} t)$, $\mathsf{cat} : p \leq \mathsf{animal} : p$, and $\mathsf{child} : p \leq \mathsf{person} : p$. Below is a small derivation in the calculus, omitting parentheses.

$$\dfrac{\dfrac{\mathsf{cat} : p \leq \mathsf{animal} : p}{\mathsf{every\ animal} : p \xrightarrow{+} t \leq \mathsf{every\ cat} : p \xrightarrow{+} t}\ (\textsc{Anti})}{\mathsf{every\ animal\ see\ some\ child} : t \leq \mathsf{every\ cat\ see\ some\ child} : t}\ (\textsc{Point})$$

In a few more steps, $\Gamma \vdash \mathsf{every\ animal\ see\ some\ child} : t \leq \mathsf{most\ cat\ see\ some\ person} : t$.

Theorem 3 (Soundness and Completeness of the Monotonicity Calculus)**.** $\Gamma \vdash s \leq t$ if and only if $\Gamma \models s \leq t$.

Our proof depends on the assumption that all base domains $\mathbb{D}_\beta$ are weakly complete (see Definition 4). This assumption entails the soundness of (WC1) and (WC2). It is open to drop (WC1) and (WC2) from the logic and also to drop the assumption of weak completeness.

Summary and Future Work

This paper adds monotonicity and antitonicity information to the typed lambda calculus, thereby providing a foundation for the Monotonicity Calculus first developed by van Benthem and others. We established properties of the types system (Definition 3), and then proposed a syntax, semantics, and proof calculus. The next steps are to add variables and λ-abstraction. There are other desirable extensions of this work of an applied nature. We also would like to have more expressive languages with completeness results.

References

[1] Harvey Friedman. Equality between functionals. In Rohit Parikh, editor, *Proceedings of Logic Colloquium '73*, volume 53 of *Lecture Notes in Mathematics*, pages 22–37, 1975.

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

[3] 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.

[4] Victor Sánchez-Valencia. *Studies on Natural Logic and Categorial Grammar.* PhD thesis, Universiteit van Amsterdam, 1991.

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

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

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

[8] Jan van Eijck. Natural logic for natural language. In Balder ten Cate and Henk Zeevat, editors, *6th International Tbilisi Symposium on Logic, Language, and Computation.* Springer, 2007.

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