Current Work on Natural Logic

and my plans for the near future

Larry Moss

NLCS, June 28, 2013

Natural logic: my take on what it's all about

Program

Re-think semantics based on computational linguistics.

Re-work the relation of logic and language, starting with inference.

First step: show that significant parts of natural language inference can be carried out in decidable logical systems.

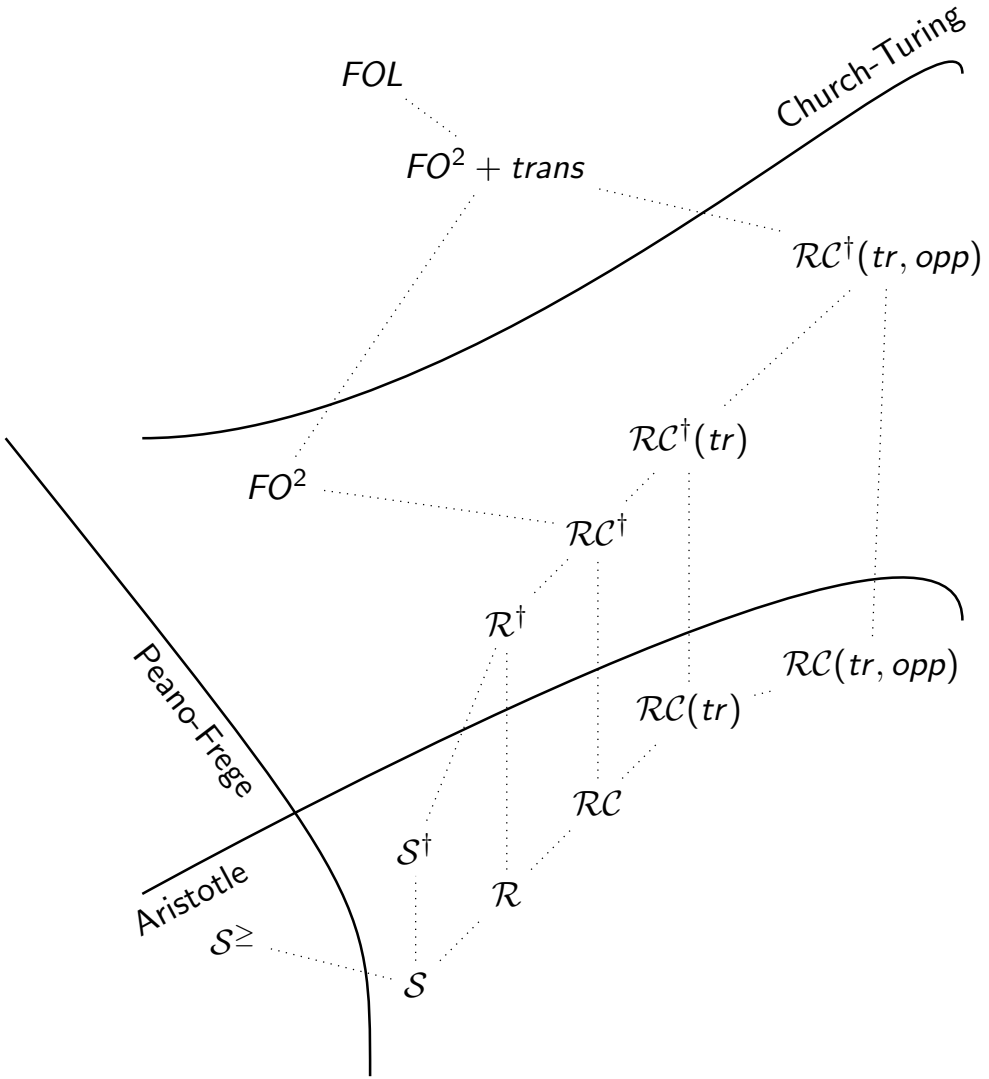
Whenever possible, to obtain complete axiomatizations, because the resulting logical systems are likely to be interesting.

To be completely mathematical and hence to work using all tools and to make connections to fields like complexity theory, (finite) model theory, proof theory, decidable fragments of first-order logic, and algebraic logic.

But these are all the first step, and they hardly touch upon the real goals.

What has been done on that first step

You have heard much of this in Ian's talk



first-order logic

FO^2 + "R is trans"

2 variable FO logic

† adds full N-negation

$\mathcal{RC}(tr)$ + opposites
$\mathcal{RC}$ + (transitive) comparative adjs
$\mathcal{R}$ + relative clauses
$\mathcal{S}$ + full N-negation
$\mathcal{R}$ = relational syllogistic
$\mathcal{S}^\geq$ adds $|p| \geq |q|$
$\mathcal{S}$: all/some/no p are q

Inference with relative clauses

What do you think about these?

$$\frac{\textit{All skunks are mammals}}{\textit{All who fear all who respect all skunks fear all who respect all mammals}}$$

$$\frac{\textit{All skunks are mammals}}{\textit{All who fear all who respect some skunks fear all who respect some mammals}}$$

$$\frac{\textit{All skunks are mammals}}{\textit{Some who fear all who respect some skunks fear some who respect some mammals}}$$

$\mathcal{RC}$ allows sentential subjects to be noun phrases containing subject relative clauses.

who r all p	*who r some p*
who don't r all p	*who don't r any p*

---

expression	syntax
$\mathcal{RC}$ sentence	$\forall(d^+, c) \mid \exists(d^+, c)$
$\mathcal{RC}^\dagger$ sentence	$\forall(d, c) \mid \exists(d, c)$

d^+ is a positive set term, and c is an arbitrary set term.

The main rules are

$$\frac{\forall(p,q)}{\forall(\forall(q,r),\forall(p,r))} \quad \frac{\forall(p,q)}{\forall(\exists(p,r),\exists(q,r))} \quad \frac{\exists(p,q)}{\forall(\forall(p,r),\exists(q,r))}$$

These rules are based on McAllester and Givan (1992).

Every giraffe is taller than every gnu
Some gnu is taller than every lion
Some lion is taller than some zebra
Every giraffe is taller than some zebra

Every giraffe is taller than every gnu
Some gnu is taller than every lion
Some lion is taller than some zebra
Every giraffe is taller than some zebra

$$\frac{\forall(p, \exists(q, r))}{\forall(\exists(p, r), \exists(q, r))} \qquad \frac{\forall(p, \forall(q, r))}{\forall(\exists(p, r), \forall(q, r))}$$

$$\frac{\exists(p, \forall(q, r))}{\forall(\forall(p, r), \forall(q, r))} \qquad \frac{\exists(p, \exists(q, r))}{\forall(\forall(p, r), \exists(q, r))}$$

Every giraffe is taller than every gnu
Some gnu is taller than every lion
Some lion is taller than some zebra
Every giraffe is taller than some zebra

$$\dfrac{\dfrac{\forall(\mathsf{gir}, \forall(\mathsf{gnu}, \mathsf{taller})) \quad \exists(\mathsf{gnu}, \forall(\mathsf{lion}, \mathsf{taller}))}{\forall(\mathsf{gir}, \forall(\mathsf{lion}, \mathsf{taller}))} \qquad \exists(\mathsf{lion}, \exists(\mathsf{zebra}, \mathsf{taller}))}{\forall(\mathsf{giraffe}, \exists(\mathsf{zebra}, \mathsf{taller}))}$$

At this time, we are starting to implement the syllogistic logics.

- GF + Haskell (Praveen Narayanen)
- miniKanren (Jason Hemann and Cameron Swords)
- C++ and the Castor Logic Paradigm

Implementation work, in their own words

Using a relation-based logic language has allowed us
to quickly and directly encode proof search algorithms for
natural logics using classic semantic models.

By encoding these semantic models in miniKanren, it is possible to generate correct-by-construction proofs of logical propositions based on the logic's rules and a set of starting premises.

While somewhat computationally intensive, this approach provides an alternative
to the standard graph-construction proof approach that has
the advantage of straightforward implementation
(as the miniKanren code closely matches the original semantic model)
and a level of generic interaction that allows us to generate proof search engines from a nearly plain-text encoding via a simple compiler.

Logic beyond the Aristotle boundary

So far in this talk, all of the systems have been syllogistic to one degree or another.

It is possible to formulate a logical system with
a restricted notion of variables,
prove completeness,
and yet stay inside the Church-Turing boundary.

Example of a proof in the system

From all keys are old items,

infer everyone who owns a key owns an old item

$$\dfrac{\dfrac{[\exists(key, own)(x)]^2 \qquad \dfrac{[own(x,y)]^1 \qquad \dfrac{[key(y)]^1 \quad \forall(key, old\text{–}item)}{old\text{–}item(y)}\ \forall E}{\exists(old\text{–}item, own)(x)}\ \exists I}{\exists(old\text{–}item, own)(x)}\ \exists E^1}{\forall(\exists(key, own), \exists(old\text{–}item, own))}\ \forall I^2$$

Example of a proof in the system

From all keys are old items,
infer everyone who owns a key owns an old item

1	$\forall(key, old\text{–}item)$	hyp
2	$\quad\exists(key, own)(x)$	hyp
3	$\quad key(y)$	$\exists E$, 2
4	$\quad own(x, y)$	$\exists E$, 2
5	$\quad old\text{–}item(y)$	$\forall E$, 1, 3
6	$\quad\exists(old\text{–}item, own)(x)$	$\exists I$, 4, 5
7	$\forall(\exists(key, own), \exists(old\text{–}item, own))$	$\forall I$, 1–6

Incorporate ideas from other logical systems:

- Nissim Francez' logic of subnectors
- Hanoch Ben-Yami 'quantified argument calculus'

Starting in on the monotonicity calculus

Assume that

$$\text{waltz} \leq \text{dance} \leq \text{move}$$

and

$$\text{grizzly} \leq \text{bear} \leq \text{animal}$$

Starting in on the monotonicity calculus

Assume that

$$\text{waltz} \leq \text{dance} \leq \text{move}$$

and

$$\text{grizzly} \leq \text{bear} \leq \text{animal}$$

Assume that we're talking about a situation where

some bears dance.

Which one would be true?

- some grizzlies dance
- some animals dance

Starting in on the monotonicity calculus

Assume that

$$\text{waltz} \leq \text{dance} \leq \text{move}$$

and

$$\text{grizzly} \leq \text{bear} \leq \text{animal}$$

Assume that we're talking about a situation where

some bears dance.

- some grizzlies dance false
- some animals dance true

Starting in on the monotonicity calculus

Assume that

$$\text{waltz} \leq \text{dance} \leq \text{move}$$

and

$$\text{grizzly} \leq \text{bear} \leq \text{animal}$$

Assume that we're talking about a situation where

some bears dance.

We write

some bears$^{\uparrow}$ dance

Now think about these two

- some bears waltz
- some bears move

Starting in on the monotonicity calculus

Assume that

$$\text{waltz} \leq \text{dance} \leq \text{move}$$

and

$$\text{grizzly} \leq \text{bear} \leq \text{animal}$$

Assume that we're talking about a situation where

some bears dance.

We write

some bears$^{\uparrow}$ dance$^{\uparrow}$

Starting in on the monotonicity calculus

Assume that

$$\text{waltz} \leq \text{dance} \leq \text{move}$$

and

$$\text{grizzly} \leq \text{bear} \leq \text{animal}$$

Put the arrows on the words bears and dance.

1. Some bears dance.
2. No bears dance.
3. Not every bear dances.
4. John sees every bear.
5. Mary sees no bear.
6. Most bears dance.
7. Any bear in Hawaii would prefer to live in Alaska.
8. If any bear dances, Harry will be happy.
9. If you play loud enough music, any bear will start to dance.
10. Doreen didn't see any bears dance in her dorm room.

The answers are:

1. Some bears$^{\uparrow}$ dance$^{\uparrow}$.
2. No bears$^{\downarrow}$ dance$^{\downarrow}$.
3. Not every bear$^{\uparrow}$ dances$^{\downarrow}$.
4. John sees every bear$^{\downarrow}$.
5. Mary sees no bear$^{\downarrow}$.
6. Most bears dance$^{\uparrow}$. (No arrow goes on bears in this one.)
7. Any bear$^{\downarrow}$ in Hawaii would prefer to live in Alaska.
8. If any bear$^{\downarrow}$ dances$^{\downarrow}$, Harry will be happy.
9. If you play loud enough music, any bear$^{\downarrow}$ will start to dance$^{\uparrow}$.
10. Doreen didn't see any bears$^{\downarrow}$ dance$^{\downarrow}$ in her dorm room.

The answers are:

1. Some bears$^{\uparrow}$ dance$^{\uparrow}$.
2. No bears$^{\downarrow}$ dance$^{\downarrow}$.
3. Not every bear$^{\uparrow}$ dances$^{\downarrow}$.
4. John sees every bear$^{\downarrow}$.
5. Mary sees no bear$^{\downarrow}$.
6. Most bears dance$^{\uparrow}$. (No arrow goes on bears in this one.)
7. Any bear$^{\downarrow}$ in Hawaii would prefer to live in Alaska.
8. If any bear$^{\downarrow}$ dances$^{\downarrow}$, Harry will be happy.
9. If you play loud enough music, any bear$^{\downarrow}$ will start to dance$^{\uparrow}$.
10. Doreen didn't see any bears$^{\downarrow}$ dance$^{\downarrow}$ in her dorm room.

The $^{\uparrow}$ and $^{\downarrow}$ notations have the same meaning in language as in math!

This is not an accident!

Let's start with an (easy) inference in algebra

Which is bigger, $-(7 + 2^{-3})$ or $-(7 + 2^{-4})$?

Let's start with an (easy) inference in algebra

Which is bigger, $-(7+2^{-3})$ or $-(7+2^{-4})$?

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

$f(x)$ monotone means if $x \leq y$, then $f(x) \leq f(y)$

$f(x)$ antitone means if $x \leq y$, then $f(y) \leq f(x)$

i.e., $f(x) \geq f(y)$

Let's start with an (easy) inference in algebra

Which is bigger, $-(7+2^{-3})$ or $-(7+2^{-4})$?

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

Throughout this talk, I'll use

blue for syntax,
and red for semantics.

Let use a type system which incorporates the monotonicity information into the types.

$$\text{plus} : r \xrightarrow{+} (r \xrightarrow{+} r) \qquad \text{minus} : r \xrightarrow{+} (r \xrightarrow{-} r)$$
$$\text{times} : r \xrightarrow{+} (r \xrightarrow{+} r) \qquad \text{div2} : r \xrightarrow{+} (r \xrightarrow{-} r)$$

To understand what this means, it will be useful to introduce the preorders corresponding to the types above, and to others.

(A preorder is a set together with a relation on it which is reflexive and transitive.)

Here is how it will work when we do the details:

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

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

Formal details on the Monotonicity Calculus

Definition

Let $\mathcal{M} = \{+, -, \cdot\}$.
We call $\mathcal{M}$ the set of markings,
and we use m to denote an element of $\mathcal{M}$.

Definition

Let $\mathcal{B}$ be a set of base types.
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 $\sigma \xrightarrow{m} \tau$, for $m \in \mathcal{M}$.

Example

In standard Montague semantics, we take $\mathcal{B}$ to be $\{e, t\}$.
In our example from algebra, we could take it to be $\{r\}$.

Example

A simple linguistic example uses base types e and t.

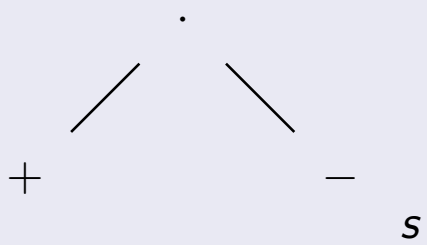
We abbreviate $e \dot{\to} t$ by p (for "property").

$$\text{every} : p \overset{-}{\to} (p \overset{+}{\to} t).$$

This reflects the facts that every is antitone in its first argument and monotone in its second.

This is the most specific type we could assign to every.

Definition ($\sqsubseteq$ on markings over arrows)



Down $\sqsubseteq$ is more specific. Up $\sqsubseteq$ is more general.

Definition ($\preceq$ on types)

$$\frac{}{\sigma \preceq \sigma} \qquad \frac{\sigma' \preceq \sigma \quad \tau \preceq \tau' \quad m \sqsubseteq m'}{\sigma \xrightarrow{m} \tau \;\preceq\; \sigma' \xrightarrow{m'} \tau'}$$

Down $\preceq$ is more specific. Up $\preceq$ is more general.

Example

Concerning every:

$$p \xrightarrow{-} (p \xrightarrow{+} t) \quad \preceq \quad p \xrightarrow{\cdot} (p \xrightarrow{\cdot} t).$$

Definition (Information order on markings)

$$m_1 \vee m_2 \quad = \quad \begin{cases} m_1 & \text{if } m_1 = m_2 \\ \cdot & \text{otherwise} \end{cases}$$

Definition

The compatibility relation $\uparrow$
and the least upper bound function $\vee$ on compatible types
are defined by recursion so that

- $\sigma \uparrow \sigma$, and $\sigma \vee \sigma = \sigma$.
- If $\tau_1 \uparrow \tau_2$, then

$$(\sigma \xrightarrow{m_1} \tau_1) \uparrow (\sigma \xrightarrow{m_2} \tau_2)$$

for all markings $m_1, m_2 \in \mathcal{M}$, and

$$(\sigma \xrightarrow{m_1} \tau_1) \vee (\sigma \xrightarrow{m_2} \tau_2) \quad = \quad \sigma \xrightarrow{m_1 \vee m_2} (\tau_1 \vee \tau_2).$$

Definition

A standard structure is a system $\mathcal{S} = \{\mathbb{D}_\tau\}_{\tau \in \mathcal{T}}$ of preorders, one for each type $\tau \in \mathcal{T}$.

For the base types $\beta \in \mathcal{B}$ there is no requirement on $\mathbb{D}_\beta$.
For complex types $\sigma \xrightarrow{m} \tau$, we have several requirements:

- $D_{\sigma \xrightarrow{+} \tau}$ is the set of all monotone functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$.
- $D_{\sigma \xrightarrow{-} \tau}$ is the set of all antitone functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$.
- $D_{\sigma \xrightarrow{\cdot} \tau}$ is the set of all functions from $\mathbb{D}_\sigma$ to $\mathbb{D}_\tau$.
- For all markings $m \in \mathcal{M}$, all types $\sigma, \tau \in \mathcal{T}$, and all $f, g \in D_{\sigma \xrightarrow{m} \tau}$, we have

$$f \leq_{\sigma \xrightarrow{m} \tau} g \text{ if and only if } f(a) \leq_\tau g(a) \text{ for all } a \in D_\sigma.$$

This is called the pointwise order.

An example of the type domains

Example

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 σ:

$\mathbb{D}_{e \xrightarrow{\cdot} t}$ will be the set of all functions from $\mathbb{D}_e$ to $\mathbb{D}_t$,
$\mathbb{D}_{e \xrightarrow{+} t}$ will be the set of monotone functions, and
$\mathbb{D}_{e \xrightarrow{-} t}$ will be the set of antitone functions.

In all cases, these are taken to be preorders using the pointwise order.

Example

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

The syntax of the calculus starts with *typed constants* $\mathsf{c} : \sigma$.

Definition

We define typed terms in the following way:

$$\frac{}{\mathsf{c} : \sigma} \qquad \frac{\mathsf{f} : \sigma \to \tau \quad \mathsf{t} : \sigma' \quad \sigma' \preceq \sigma}{\mathsf{f}(\mathsf{t}) : \tau}$$

This ability to apply a function symbol of type $\sigma \to \tau$
to an argument of more specific type $\sigma' \preceq \sigma$
is a design feature of our approach.

Example

For an example pertaining to algebra, we take

$$\mathsf{plus} : r \xrightarrow{+} (r \xrightarrow{+} r) \qquad \mathsf{minus} : r \xrightarrow{+} (r \xrightarrow{-} r)$$
$$\mathsf{times} : r \xrightarrow{+} (r \xrightarrow{+} r) \qquad \mathsf{div2} : r \xrightarrow{+} (r \xrightarrow{-} r)$$

and we might as well add several more symbols

$$\mathsf{abs} : r \xrightarrow{\cdot} r$$
$$0, 1, 2 : r$$

Typed constants: example pertinent to natural language.

Example

We take plural nouns like cat, person, ... : p.
Also, we take determiners

$$\left.\begin{array}{l} \text{every} : p \xrightarrow{-} (p \xrightarrow{+} t) \\ \text{not every} : p \xrightarrow{+} (p \xrightarrow{-} t) \\ \text{some} : p \xrightarrow{+} (p \xrightarrow{+} t) \\ \text{no} : p \xrightarrow{-} (p \xrightarrow{-} t) \\ \text{most} : p \xrightarrow{\cdot} (p \xrightarrow{+} t) \\ \text{exactly } n : p \xrightarrow{\cdot} (p \xrightarrow{\cdot} t) \end{array}\right\}$$

These types are related by $\uparrow$ pairwise.

This is another design feature of our approach!

Intransitive and transitive verbs (e.g. vomits, see)

$$\text{vomits} : p \qquad \text{see} : (p \xrightarrow{+} t) \xrightarrow{+} t.$$

Denotation of terms in a standard structure

Definition

For each term $t : \tau$, and for each $\tau' \succeq \tau$, we define $[\![t]\!]_{\tau'}$:

- The semantics begins with values $[\![c]\!]_\tau$.

  We require that $[\![c]\!]_\tau$ belong to $\mathbb{D}_\tau$.
- If $t : \sigma \xrightarrow{m} \tau$ and $u : \sigma'$ with $\sigma' \preceq \sigma$, then

$$[\![t(u)]\!]_\tau = [\![t]\!]_{\sigma \xrightarrow{m} \tau}\big([\![u]\!]_{\sigma'}\big).$$

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

Frequently we omit the subscripts on the types.

Example

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

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

$[\![\mathsf{c}]\!] \in \mathbb{D}_\sigma$ for constants $\mathsf{c} : \sigma$.
That is, we have a bona fide semantics of all constants.

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

$$[\![\mathsf{plus\ 1\ 1}]\!] \quad = \quad [\![\mathsf{plus}]\!]([\![\mathsf{1}]\!])([\![\mathsf{1}]\!]) \quad = \quad [\![\mathsf{plus}]\!](1)(1) \quad = \quad 2.$$

Semantics of inequalities in structures

Definition (Satisfaction in a standard structure)

$$\mathcal{S} \models s \leq t \text{ means that } [\![s]\!] \leq [\![t]\!] \text{ in } \mathbb{D}_{\sigma \vee \tau}$$

We only use this notation
when the types of s and t are compatible ($\uparrow$),
and we are comparing their values inside $\mathbb{D}_{\sigma \vee \tau}$.

Definition (Semantic consequence)

We always use Γ to denote a set of inequalities of the form $u \leq v$.

We write $\Gamma \models s \leq t$ to mean

> whenever $\mathcal{S} \models u \leq v$ for all inequalities $u \leq v \in \Gamma$,
> also $\mathcal{S} \models s \leq t$.

Labeling subterms inside of terms

Suppose u is a subterm occurrence in t.
We define $l \in \mathcal{M}$ 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:

- If $u = t$, then $u[u^{\uparrow}]$;
- If $s[u^l]$, then $s(v)[u^l]$;
- If $v[u^l]$ and $s : \tau \xrightarrow{m} \sigma$, then $s(v)[u^{m \circ l}]$.

You can use your favorite algorithm.

Example

see every $[\text{cat}^{\downarrow}]$
see some $[\text{cat}^{\uparrow}]$
see most $[\text{cat}^{\cdot}]$

Example

As we have seen

div2 minus $[x^{\uparrow}]$ $[y^{\downarrow}]$ minus $[z^{\downarrow}]$ plus $[v^{\uparrow}]$ $[w^{\uparrow}]$.

The Monotonicity Calculus

I have omitted the types

$$\frac{}{t \leq t}\ (\textsc{Reflexive}) \qquad \frac{t \leq u \quad u \leq v}{t \leq v}\ (\textsc{Transitive})$$

$$\frac{f \leq g}{f(t) \leq g(t)}\ (\textsc{Pointwise})$$

$$\frac{u \leq v}{t[u^{\uparrow}] \leq t^{v \leftarrow u}}\ (\textsc{Monotone}) \qquad \frac{u \leq v}{t[v^{\downarrow}] \leq t^{u \leftarrow v}}\ (\textsc{Antitone})$$

Soundness/Completeness Theorem

$$\Gamma \vdash t \leq u \quad \text{iff} \quad \Gamma \models t \leq u.$$

Actually, I'm cheating here.

Our current proof needs to assume that our preorders are weakly complete:
every two elements have an upper bound and a lower bound.

And we also have two extra rules in the calculus, unfortunately.

$$\frac{f^{\downarrow} \leq g^{\uparrow}}{f^{\downarrow}(a) \leq g^{\uparrow}(b)}\ (\text{WC1}) \qquad \frac{f^{\uparrow} \leq g^{\downarrow}}{f^{\uparrow}(a) \leq g^{\downarrow}(b)}\ (\text{WC2})$$

EXAMPLE

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

$$\dfrac{\dfrac{\dfrac{1 \leq 2}{\mathsf{minus}\ [1^{\uparrow}]\ \leq \mathsf{minus}\ 2}\ (\textsc{Mono})}{\mathsf{minus}\ 1\ 1 \leq \mathsf{minus}\ 2\ 1}\ (\textsc{Point}) \quad \dfrac{0 \leq 1}{\mathsf{minus}\ 2\ [1^{\downarrow}] \leq \mathsf{minus}\ 2\ 0}\ (\textsc{Anti})}{\mathsf{minus}\ 1\ 1 \leq \mathsf{minus}\ 2\ 0}\ (\textsc{Trans})$$

Example

Let Γ contain the following inequalities

$$\begin{array}{lcl} \mathsf{cat} : p & \leq & \mathsf{animal} : p \\ \mathsf{every} : p \xrightarrow{-} (p \xrightarrow{+} t) & \leq & \mathsf{most} : p \xrightarrow{\cdot} (p \xrightarrow{+} t) \end{array}$$

Recall that the types of every and most are related by $\uparrow$.
So it makes sense for us to write every $\leq$ most.
It would not be sensible to write every $\leq$ cat.

Part of our contribution is exactly settling this kind of issue.

Example

Let Γ contain the following inequalities

$$\begin{array}{lcl} \mathsf{cat} : p & \leq & \mathsf{animal} : p \\ \mathsf{every} : p \xrightarrow{-} (p \xrightarrow{+} t) & \leq & \mathsf{most} : p \xrightarrow{\cdot} (p \xrightarrow{+} t) \end{array}$$

Below is a derivation from Γ:

$$\dfrac{\dfrac{\dfrac{\mathsf{cat} \leq \mathsf{animal}}{\mathsf{every}\ [\mathsf{animal}^{\downarrow}] \leq \mathsf{every\ cat}}\ (\textsc{Anti}) \quad \dfrac{\mathsf{every} \leq \mathsf{most}}{\mathsf{every\ cat} \leq \mathsf{most\ cat}}\ (\textsc{Point})}{\mathsf{every\ animal} \leq \mathsf{most\ cat}}\ (\textsc{Trans})}{\mathsf{every\ animal\ vomits} \leq \mathsf{most\ cat\ vomits}}\ (\textsc{Point})$$

Combining the arithmetic and linguistic examples

Example

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

$$\text{at least} : r \xrightarrow{-} (p \xrightarrow{+} (p \xrightarrow{+} t))$$
$$\text{at most} : r \xrightarrow{+} (p \xrightarrow{-} (p \xrightarrow{-} t))$$
$$\text{more than} : r \xrightarrow{-} (p \xrightarrow{+} (p \xrightarrow{+} t))$$
$$\text{less than} : r \xrightarrow{+} (p \xrightarrow{-} (p \xrightarrow{-} t))$$

Then the natural set Γ of assumptions would include

$$0 \leq 1, 1 \leq 2, \ldots, \text{more than} \leq \text{at least},$$
$$\text{less than} \leq \text{at most}, \quad \text{some} \leq \text{at least 1}, \quad \text{at least 1} \leq \text{some}$$

For example, we could prove

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

Next steps in this line of work

The next step would be to add λ-abstraction
and to incorporate the α, β, and η rules of the Lambda Calculus.

What we have done already is like the Lambda Calculus
but with constants only, no abstraction.

In the typed Lambda Calculus, we study equational reasoning
concerning equalities $\mathtt{t} = \mathtt{u}$ between terms of the same type.

Work on the constants is more interesting for the
Monotonicity Calculus than the simply-typed Lambda Calculus.