Monotonicity Calculus: What is for, and How it is Formulated

Larry Moss

Indiana University

Nordic Logic School
August 7-11, 2017

What I plan to do is re-do the material on the $\uparrow$ and $\downarrow$ notation that you just on Monday,
but very slowly,
and to aim for the formal work.

We have already, in effect, seen this

Assume that

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

and

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

We also have:

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.

So far, our treatment of the $\uparrow$and $\downarrow$notation has been purely suggestive:

- We didn't really explain why any of it worked.
- We didn't even say how it worked.
- We didn't explain where the assumptions came from.

The primary sources on this topics are a paper and book chapter by Johan van Benthem (1986)
and the dissertation of Victor Sanchez Valencia (1991).

1. Phil didn't dream of any monsters.
2. *Phil dreamt of any monsters.
3. Daisy wondered whether Phil dreamt of any monsters.
4. Daisy doubted that Phil dreamt of any monsters.
5. *Daisy believed that Phil dreamt of any monsters.

1. Phil didn't dream of any monsters.
2. *Phil dreamt of any monsters.
3. Daisy wondered whether Phil dreamt of any monsters.
4. Daisy doubted that Phil dreamt of any monsters.
5. *Daisy believed that Phil dreamt of any monsters.

Facts about the distribution and meaning of any
and other negative polarity arguments
are partly syntactic, partly semantic, and partly
pragmatic.

(So the whole study is very complicated!)

We only will touch on the semantic side of things.

What do the various $\uparrow$ and $\downarrow$
judgments have in common?

Can we automatically determine the $\uparrow$ and $\downarrow$ notations
(in parse trees according to some grammar)?

How does inference connect to $\uparrow$ and $\downarrow$?

What can we say about negative polarity items?

Where we're going in this talk

- A parallel: elementary mathematics
- Categorial grammar and semantics
- Mathematical inference as grammatical inference
- Back to natural language grammars
- Monotonicity calculus
- A completeness theorem

Definition

A function $f(x)$ is monotone if

$$x \leq y \quad \text{implies} \quad f(x) \leq f(y)$$

We also indicate this by writing

$$\frac{x \leq y}{f(x) \leq f(y)}\ \mathrm{f(x)}^{\uparrow}$$

Definition

A function $f(x)$ is antitone if

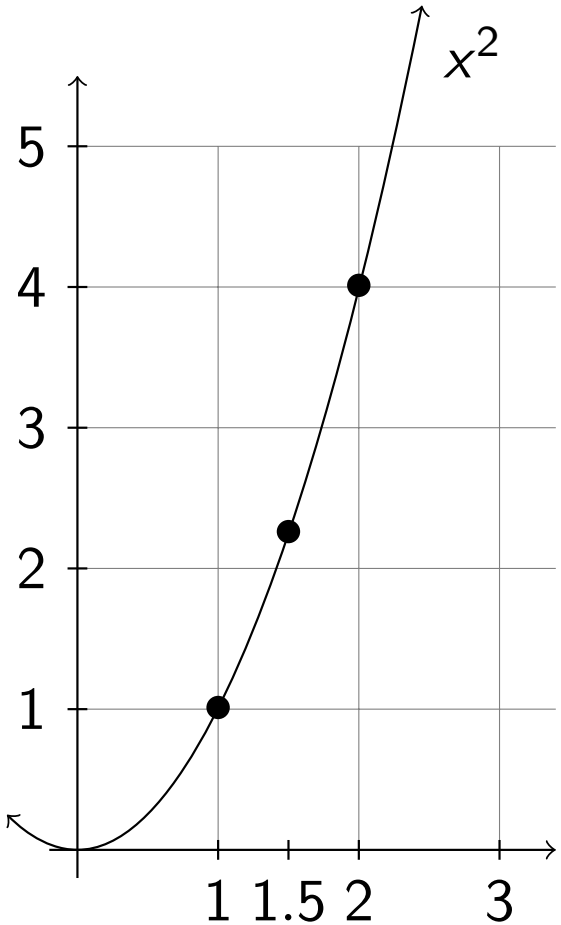
$$x \leq y \quad \text{implies} \quad f(y) \leq f(x)$$

We also indicate this by writing

$$\frac{x \leq y}{f(y) \leq f(x)}\ \mathrm{f(x)}^{\downarrow}$$

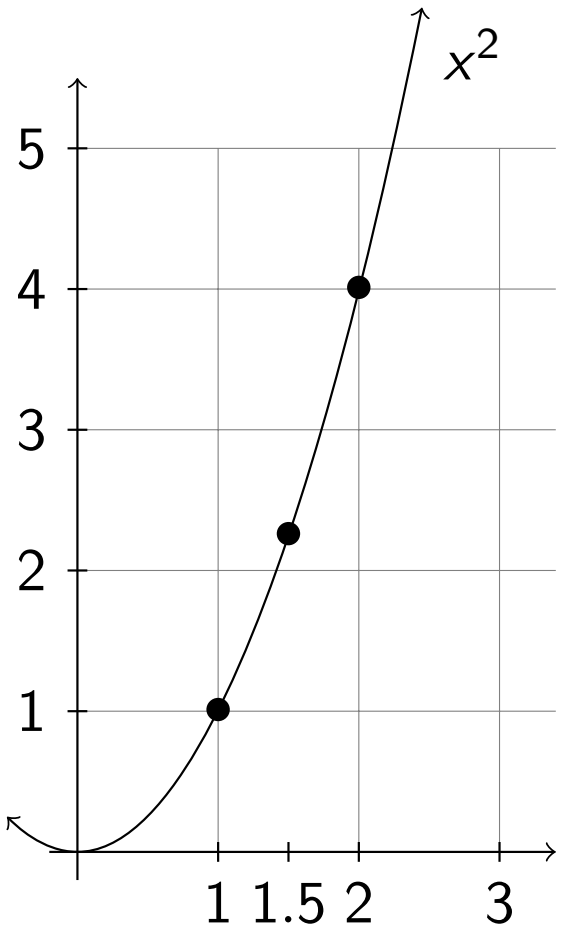
$f(x) = x^2$ is increasing on $[0, \infty]$

Bigger inputs give bigger outputs



$f(x) = x^2$ is increasing on $[0, \infty]$

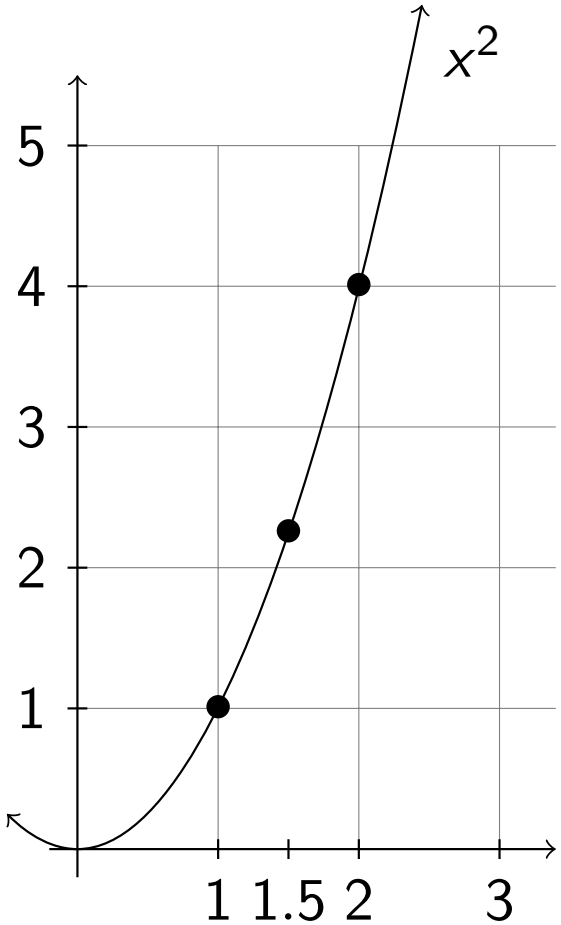
Bigger inputs give bigger outputs



Since $1 < 1.5$, $1^2 \leq (1.5)^2$.

$f(x) = x^2$ is increasing on $[0, \infty]$

Bigger inputs give bigger outputs



Since $1 < 1.5$, $1^2 \leq (1.5)^2$.
Since $1.5 < 2$, $(1.5)^2 \leq 4$.

Use $\uparrow$ for *increasing* and $\downarrow$ for *decreasing*.

$f(x)$	$\uparrow$ or $\downarrow$ on $[0, \infty]$
x	
$-x$	
$7 + x$	
x^2	
$1/x$	
2^x	
2^{-x}	

Use $\uparrow$ for *increasing* and $\downarrow$ for *decreasing*.

$f(x)$	$\uparrow$ or $\downarrow$ on $[0, \infty]$
x	$\uparrow$
$-x$	$\downarrow$
$7 + x$	$\uparrow$
x^2	$\uparrow$
$1/x$	$\downarrow$
2^x	$\uparrow$
2^{-x}	$\downarrow$

Example of reasoning with increasing and decreasing functions

Here's a harder problem.

Tell which is bigger without a calculator:

$$\left(\frac{29}{4}\right)^{-3} \quad \text{or} \quad \left(7+\frac{1}{\pi^2}\right)^{-3}$$

This will be a little easier if we write $29/4$ as an improper fraction:

$$\left(7+\frac{1}{4}\right)^{-3} \quad \text{vs.} \quad \left(7+\frac{1}{\pi^2}\right)^{-3}$$

Example of reasoning with increasing and decreasing functions

Here's a harder problem.

This will be a little easier if we write 29/4 as an improper fraction:

$$\left(7+\frac{1}{4}\right)^{-3} \quad \text{vs.} \quad \left(7+\frac{1}{\pi^2}\right)^{-3}$$

$$\dfrac{\dfrac{\dfrac{2 \le \pi}{\frac{1}{\pi} \le \frac{1}{2}}\ 1/x\downarrow}{\frac{1}{\pi^2} \le \frac{1}{4}}\ x^2\uparrow}{7+\frac{1}{\pi^2} \le 7+\frac{1}{4}}\ 7+x\uparrow$$

$$\dfrac{}{(7+\frac{1}{4})^{-3} \le (7+\frac{1}{\pi^2})^{-3}}\ x^{-3}\downarrow$$

Tell which is bigger without a calculator:

$$\left(-\frac{7}{4}\right)^{-3} \quad \text{or} \quad \left(-\frac{7}{\pi}\right)^{-3}$$

Do you remember composition of functions?

Suppose that

$$v(x) = 1/x \qquad u(x) = x^2 \qquad t(x) = 7 + x \qquad s(x) = x^{-3}$$

What is $v \circ u$?
What is $s \circ t$?
What is $s \circ (t \circ v)$?

Do you remember composition of functions?

Suppose that

$$v(x) = 1/x \qquad u(x) = x^2 \qquad t(x) = 7 + x \qquad s(x) = x^{-3}$$

Let's write

$$\left(7 + \frac{1}{x^2}\right)^{-3}$$

as a composition of these four functions?

Do you remember composition of functions?

Suppose that

$$v(x) = 1/x \qquad u(x) = x^2 \qquad t(x) = 7 + x \qquad s(x) = x^{-3}$$

Let's write

$$\left(7 + \frac{1}{x^2}\right)^{-3}$$

as a composition of these four functions?

It is either one of the following

$$s \circ t \circ u \circ v$$
$$s \circ t \circ v \circ u$$

Composition of functions and increasing/decreasing functions

If f is increasing and g is increasing, then $g \circ f$ is increasing.

If f is increasing and g is decreasing, then $g \circ f$ is decreasing.

If f is decreasing and g is increasing, then $g \circ f$ is decreasing.

If f is decreasing and g is decreasing, then $g \circ f$ is increasing.

If x is positive and y is positive, then xy is positive.

If x is positive and y is negative, then xy is negative.

If x is negative and y is positive, then xy is negative.

If x is negative and y is negative, then xy is positive.

The analogy is

$$\begin{array}{lcl} \text{increasing} & \approx & \text{positive} \\ \text{decreasing} & \approx & \text{negative} \end{array}$$

Another way to see that

$$\left(7 + \frac{1}{4}\right)^{-3} \quad \leq \quad \left(7 + \frac{1}{\pi^2}\right)^{-3}$$

Consider the function $s \circ t \circ u \circ v$, where

$$v(x) = 1/x \qquad u(x) = x^2 \qquad t(x) = 7 + x \qquad s(x) = x^{-3}$$

Another way to see that

$$\left(7 + \frac{1}{4}\right)^{-3} \quad \leq \quad \left(7 + \frac{1}{\pi^2}\right)^{-3}$$

We just saw that $f(x)$ can be written as $s \circ t \circ u \circ v$, where

$$v(x) = 1/x \quad u(x) = x^2 \quad t(x) = 7 + x \quad s(x) = x^{-3}$$

$$\downarrow \qquad \uparrow \qquad \uparrow \qquad \downarrow$$

Another way to see that

$$\left(7 + \frac{1}{4}\right)^{-3} \quad \leq \quad \left(7 + \frac{1}{\pi^2}\right)^{-3}$$

We just saw that $f(x)$ can be written as $s \circ t \circ u \circ v$, where

$$v(x) = 1/x \qquad u(x) = x^2 \qquad t(x) = 7 + x \qquad s(x) = x^{-3}$$
$$\downarrow \qquad\qquad \uparrow \qquad\qquad \uparrow \qquad\qquad \downarrow$$

Since the number of $\downarrow$s is even, our function $f(x)$ is $\uparrow$, increasing.

So since $2 \leq \pi$, we have $f(2) \leq f(\pi)$.

This is what we want to show.

Consider

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

Suppose we fix numerical values for the variables v, w, x, y, z, and then suppose we also increase x a bit to some number $x' \geq x$.

Does the value of f go up or down?

That is, which is true:

$$\begin{array}{ll} \text{if } x \leq x', \text{ then} & f(v, w, x, y, z) \leq f(v, w, x', y, z) \\ \text{if } x \leq x', \text{ then} & f(v, w, x', y, z) \leq f(v, w, x, y, z) \end{array}$$

A moment's though shows that the first line is correct.

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

Next, suppose we fix values but this time move y up to some number y'.

Which is true now?

$$\begin{array}{ll} \text{if } x \leq x', \text{ then} & f(v, w, x, y, z) \leq f(v, w, x', y, z) \\ \text{if } x \leq x', \text{ then} & f(v, w, x', y, z) \leq f(v, w, x, y, z) \end{array}$$

This time, it's the second one.

We also can study z, v, and w the same way.

We would summarize all of the observations by writing

$$f(v^{\uparrow}, w^{\uparrow}, x^{\uparrow}, y^{\downarrow}, z^{\downarrow}).$$

Here is how we can think about this in terms close to the way formal semantics works with the simply typed Lambda Calculus.

We take a single type r, and then we have function symbols

$$\text{plus} : r \to (r \to r) \qquad \text{minus} : r \to (r \to r)$$
$$\text{times} : r \to (r \to r) \qquad \text{div2} : r \to (r \to r)$$

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

Here is how we can think about this in terms close to the way formal semantics works with the simply typed Lambda Calculus.

We take a single type r, and then we have function symbols

$$\text{plus} : r \to (r \to r) \qquad \text{minus} : r \to (r \to r)$$
$$\text{times} : r \to (r \to r) \qquad \text{div2} : r \to (r \to r)$$

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

NB. The semantics will use higher-order (one-place) functions.

We get terms in Polish notation

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

This would correspond to the term usually written $z - (v + w)$.

We get terms in Polish notation

We are interested in a term corresponding to

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

To fit it all on the screen, let's drop the types:

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

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

$\mathsf{div2}(\mathsf{t})(\mathsf{u})$ is supposed to mean $2^{t \div u}$.

Can we determine the polarities of the variables from the tree?

Go from the root to the leaves, marking

green for $\uparrow$ red for $\downarrow$

The rule for propagating colors is:

the right branches of "completed" nodes for *div2* and *minus* flip colors.
Otherwise, we keep colors as we go up the tree.

Can we determine the polarities of the variables from the tree?

Go from the root to the leaves, marking

green for $\uparrow$ red for $\downarrow$

The rule for propagating colors is:

the right branches of "completed" nodes for
div2 and *minus* flip colors.
Otherwise, we keep colors as we go up the tree.

$$\dfrac{\dfrac{\textit{div2}\quad \dfrac{\textit{minus}\quad x}{\textit{minus}\ x\quad y}}{\textit{div2}\ \textit{minus}\ x\ y}\qquad \dfrac{\dfrac{\textit{minus}\quad z}{\textit{minus}\ z}\quad \dfrac{\textit{plus}\quad v}{\textit{plus}\ v\quad w}}{\textit{plus}\ v\ w}}{\textit{minus}\ z\ \textit{plus}\ v\ w}}{\textit{div2}\ \textit{minus}\ x\ y\ \textit{minus}\ z\ \textit{plus}\ v\ w}$$

Can we determine the polarities of the variables from the tree?

Go from the root to the leaves, marking

green for $\uparrow$ red for $\downarrow$

The rule for propagating colors is:

the right branches of "completed" nodes for
div2 and *minus* flip colors.
Otherwise, we keep colors as we go up the tree.

$$\dfrac{\dfrac{\textit{div2} \quad \dfrac{\textit{minus} \quad x}{\textit{minus}\ x \quad y}}{\textit{minus}\ x\ y} \qquad \dfrac{\dfrac{\textit{minus} \quad z}{\textit{minus}\ z} \qquad \dfrac{\textit{plus} \quad v}{\textit{plus}\ v \quad w}}{\textit{plus}\ v\ w}}{\textit{div2}\ \textit{minus}\ x\ y \qquad \textit{minus}\ z\ \textit{plus}\ v\ w}$$

$$\textit{div2}\ \textit{minus}\ x\ y\ \textit{minus}\ z\ \textit{plus}\ v\ w$$

Can we determine the polarities of the variables from the tree?

Go from the root to the leaves, marking

green for $\uparrow$ red for $\downarrow$

The rule for propagating colors is:

the right branches of "completed" nodes for
div2 and *minus* flip colors.
Otherwise, we keep colors as we go up the tree.

$$\dfrac{\dfrac{\dfrac{\dfrac{\textit{minus}\quad x}{\textit{minus}\ x\quad y}}{\textit{div2}\quad \textit{minus}\ x\ y} \qquad \dfrac{\dfrac{\textit{minus}\quad z}{\textit{minus}\ z}\qquad \dfrac{\dfrac{\textit{plus}\quad v}{\textit{plus}\ v\quad w}}{\textit{plus}\ v\ w}}{\textit{minus}\ z\ \textit{plus}\ v\ w}}{\textit{div2}\ \textit{minus}\ x\ y\ \textit{minus}\ z\ \textit{plus}\ v\ w}$$

Can we determine the polarities of the variables from the tree?

Go from the root to the leaves, marking

green for $\uparrow$ red for $\downarrow$

The rule for propagating colors is:

the right branches of "completed" nodes for
div2 and *minus* flip colors.
Otherwise, we keep colors as we go up the tree.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathit{minus}\quad x}\quad \dfrac{}{\mathit{minus}\ x\quad y}}{\mathit{div2}\quad \mathit{minus}\ x\ y}\quad \dfrac{\dfrac{}{\mathit{minus}\quad z}\quad \dfrac{}{\mathit{plus}\quad v}}{\mathit{minus}\ z}\quad \dfrac{}{\mathit{plus}\ v\quad w}}{\mathit{plus}\ v\ w}}{\mathit{div2}\ \mathit{minus}\ x\ y\quad \mathit{minus}\ z\ \mathit{plus}\ v\ w}}{\mathit{div2}\ \mathit{minus}\ x\ y\ \mathit{minus}\ z\ \mathit{plus}\ v\ w}$$

Can we determine the polarities of the variables from the tree?

Go from the root to the leaves, marking

green for $\uparrow$ red for $\downarrow$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathit{minus}\quad x}}{\mathit{minus}\ x\quad y}}{\mathit{div2}\quad \mathit{minus}\ x\ y}}{\mathit{div2}\ \mathit{minus}\ x\ y} \quad \dfrac{\dfrac{\dfrac{}{\mathit{minus}\quad z}}{\mathit{minus}\ z}}{\mathit{minus}\ z} \quad \dfrac{\dfrac{\dfrac{}{\mathit{plus}\quad v}}{\mathit{plus}\ v\quad w}}{\mathit{plus}\ v\ w}}{\mathit{minus}\ z\ \mathit{plus}\ v\ w}}{\mathit{div2}\ \mathit{minus}\ x\ y\ \mathit{minus}\ z\ \mathit{plus}\ v\ w}$$

This agrees with what we saw before:

$$f(v^{\uparrow}, w^{\uparrow}, x^{\uparrow}, y^{\downarrow}, z^{\downarrow}).$$

This algorithm has a history

It was first proposed in CG by van Benthem in the 1990's
to formalize the $\uparrow$, $\downarrow$ notation.

His proposal was then worked out by Sanchez-Valencia.

(Older versions exists: e.g., Sommers.)

Versions of it are even implemented in
real-world CL systems:

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.

This makes the determination of polarities an external feature of the syntax tree,
something determined by an algorithm.

Instead of complicating the architecture of grammar.
let's complicating the particular grammar that we use.

But before we do that,
I want to go over how simple types work,
and how we can re-work them.

Review of the architecture of types and lexicons

We begin with a set $\mathcal{T}_0$ of basic types.

Then if σ and τ are types, so is (σ, τ).
(Today I'll write often this as $\sigma \to \tau$.)

The set of all types is written $\mathcal{T}$.

Review of the architecture of types and lexicons

We begin with a set $\mathcal{T}_0$ of basic types.

Then if σ and τ are types, so is (σ, τ).
(Today I'll write often this as $\sigma \to \tau$.)

The set of all types is written $\mathcal{T}$.

Let D be a function which assigns sets to the basic types.

Frequently, the basic types in NL semantics are e and t,
so we would have $D(e)$ and $D(t)$.

We would write these as D_e and D_t, since this is what everyone does.

Review of the architecture of types and lexicons

We begin with a set $\mathcal{T}_0$ of basic types.

Then if σ and τ are types, so is (σ, τ).
(Today I'll write often this as $\sigma \to \tau$.)

The set of all types is written $\mathcal{T}$.

But in our arithmetic example,
there is only one basic type: r.

And our natural choice for D_r is the set of real numbers.

Review of the architecture of types and lexicons

We begin with a set $\mathcal{T}_0$ of basic types.

Then if σ and τ are types, so is (σ, τ).
(Today I'll write often this as $\sigma \to \tau$.)

The set of all types is written $\mathcal{T}$.

Let D be a function which assigns sets to the basic types.

This function D can be extended to all types by the rule

$$D_{(\sigma \to \tau)} \quad = \quad (D_\sigma \to D_\tau)$$

This is the set of all functions from D_σ to D_τ.

Given a syntactic item in a lexicon like

$$(word, \sigma)$$

our semantics must assign to w some interpretation

$$[\![\text{word}]\!] \in D_\sigma$$

It then follows that if T is a all parse tree over the given lexicon and the root of T is $(junk, \sigma)$,
then $[\![\text{junk}]\!]$ will belong to D_σ.

We said that each D_σ was a set,
and now we want it to be a preorder.

So we'll usually write it as $\mathbb{P}_\sigma$.

We previously said

$$D_{(\sigma\tau)} \quad = \quad (D_\sigma \to D_\tau)$$

This is the set of all functions from D_σ to D_τ.

But now we have to re-think this because we want
preorders, not just sets.

The natural thing to do is to use the pointwise order

$$\mathbb{P}_{(\sigma\to\tau)} \quad = \quad (\mathbb{P}_\sigma \to \mathbb{P}_\tau)$$

This is the set of all functions from $\mathbb{P}_\sigma$ to $\mathbb{P}_\tau$, ordered by

$$f \leq g \quad \text{iff} \quad \text{for all } x \in \mathbb{P}_\sigma,\ f(x) \leq g(x) \text{ in } \mathbb{P}_\tau.$$

But here we have a problem: we sometimes want antitone functions to handle all the $\downarrow$ stuff.

Two choices: here's the first

One way would be to change our type system:

- If σ and τ are types, so are $(\sigma, \tau)^+$ and $(\sigma, \tau)^-$.

Then define

$$\mathbb{P}_{(\sigma,\tau)^+} \quad = \quad (\mathbb{P}_\sigma \to \mathbb{P}_\tau)^+$$

This is the set of all monotone functions from $\mathbb{P}_\sigma$ to $\mathbb{P}_\tau$, ordered by

$$f \leq g \quad \text{iff} \quad \text{for all } x \in \mathbb{P}_\sigma,\ f(x) \leq g(x) \text{ in } \mathbb{P}_\tau.$$

$$\mathbb{P}_{(\sigma,\tau)^-} \quad = \quad (\mathbb{P}_\sigma \to \mathbb{P}_\tau)^-$$

This is the set of all antitone functions from $\mathbb{P}_\sigma$ to $\mathbb{P}_\tau$, ordered by (presumably)

$$f \leq g \quad \text{iff} \quad \text{for all } x \in \mathbb{P}_\sigma,\ f(x) \leq g(x) \text{ in } \mathbb{P}_\tau.$$

- If σ and τ are types, so are $-\sigma$ and also (σ, τ).

Then define

$$\mathbb{P}_{-\sigma} \quad = \quad -\mathbb{P}_{\sigma}$$

This is the opposite order of $\mathbb{P}_{\sigma}$.

And

$$\mathbb{P}_{\sigma\to\tau} \quad = \quad (\mathbb{P}_{\sigma} \to \mathbb{P}_{\tau})$$

This is the set of all monotone functions from $\mathbb{P}_{\sigma}$ to $\mathbb{P}_{\tau}$,

Let us expand on that type assignment by incorporating the monotonicity information into the types.

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

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

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

OLD

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

NEW

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

Old Algorithm

- Label the root with $\uparrow$.
- Propagate notations up the tree.
  The right branches of nodes for div2 and minus of type r flip notations.
  Otherwise, we maintain the notations as we go up the tree.

New Algorithm

- Label the root with $\uparrow$.
- Propagate notations up the tree.
  - If a node is labeled ℓ and its left parent is of type $\sigma \xrightarrow{+} \tau$, then both parents are labeled ℓ.
  - If a node is labeled ℓ and its left parent is of type $\sigma \xrightarrow{-} \tau$, then the left parent is to be labeled ℓ and the right parent is to be labeled with ℓ turned upside-down.

Top-down version of the algorithm

- Label one of the leaves with $\uparrow$.
- Propagate notations down the tree.
  - If a node is labeled ℓ and it is a functor node, then its child down the tree gets the same label ℓ.
  - If a node n is labeled ℓ and n is an argument node,
    then: if the type of the functor node m is $\sigma \xrightarrow{+} \tau$,
    again the child of n gets the label ℓ.
    But if the type of the functor node m is $\sigma \xrightarrow{-} \tau$, then the child of n gets the label ℓ turned upside-down.

Lexicon

$v : r$	$v : -r$
$w : r$	$w : -r$
$x : r$	$x : -r$
$y : r$	$y : -r$
$z : r$	$z : -r$
$plus : r \to (r \to r)$	$plus : -r \to (-r \to -r)$
$minus : r \to (-r \to r)$	$minus : -r \to (r \to -r)$
$times : r \to (r \to r)$	$times : -r \to (-r \to -r)$
$div2 : r \to (-r \to r)$	$div2 : -r \to (r \to -r)$

The left side comes from the raw facts of arithmetic, the right side from a formal duality based on

$$\begin{aligned} -(s \to t) &= -s \to -t \\ -(-s) &= s \end{aligned}$$

.

Lexicon

$plus : r \to (r \to r)$	$plus : -r \to (-r \to -r)$
$minus : r \to (-r \to r)$	$minus : -r \to (r \to -r)$
$times : r \to (r \to r)$	$times : -r \to (-r \to -r)$
$div2 : r \to (-r \to r)$	$div2 : -r \to (r \to -r)$

Now a term corresponding to

$$\frac{z}{2^{x-y}}$$

parses as

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

The parse tree automatically indicates the polarities.

It will be much simpler if we replace "variables" in our syntax by actual numbers:

Lexicon

$1 : r$	$1 : -r$
$2 : r$	$2 : -r$
$3 : r$	$3 : -r$
$plus : r \to (r \to r)$	$plus : -r \to (-r \to -r)$
$minus : r \to (-r \to r)$	$minus : -r \to (r \to -r)$
$times : r \to (r \to r)$	$times : -r \to (-r \to -r)$
$div2 : r \to (-r \to r)$	$div2 : -r \to (r \to -r)$

Our only basic type is r.
Take $\mathbb{P}_r$ = usual real numbers with the usual order.

Then $\mathbb{P}_{-r} = -\mathbb{P}_r$ = usual real numbers with the opposite order.

$$\begin{array}{lcl}
\mathbb{P}_{r\to r} & = & \text{monotone functions from } R \text{ to } R \\
\mathbb{P}_{-r\to r} & = & \text{monotone functions from } -R \text{ to } R \\
\mathbb{P}_{r\to -r} & = & \text{monotone functions from } R \text{ to } -R \\
\mathbb{P}_{-r\to -r} & = & \text{monotone functions from } -R \text{ to } -R
\end{array}$$

We need interpretations for *plus*, *minus*, *times*, *div*2.
Actually, we need *two* interpretations for each.

For $plus : r \to (r \to r)$, we take

$$[\![\text{plus}]\!] : R \to R^R$$

which takes each real number a to:
the function which takes each real number b to $a + b$.

For example, $[\![\text{plus}]\!](67)(-3) = 64$.

We have to check that $[\![\text{plus}]\!]$ really belongs to $D_{r\to(r\to r)}$.

- This means: For each a, $[\![\text{plus}]\!](a)$ is a monotone function: If $b \leq b'$, then $a + b \leq a + b'$.
- The function from R to $(R \to R)$ taking a to $[\![\text{plus}]\!](a)$ is itself monotone.
  This means that if $a \leq a'$, then for all b, $a + b \leq a' + b$.

We also have to interpret $\mathit{plus} : -r \to (-r \to -r)$.

Fact

$-(\mathbb{P}, \mathbb{Q}) = (-\mathbb{P}, -\mathbb{Q})$.

Using this,

$$\mathbb{P}_{(-r\to(-r\to-r))} \quad = \quad -\mathbb{P}_{(r\to(r\to r))}.$$

Fact

A preorder and its opposite have the same set of points.

So we automatically see that $[\![\text{plus}]\!]$ belongs to $\mathbb{P}_{(-r\to(-r\to-r))}$.

We similarly use

$$\begin{aligned}
[\![\text{plus}]\!](a)(b) &= a + b \\
[\![\text{minus}]\!](a)(b) &= a - b \\
[\![\text{times}]\!](a)(b) &= a \times b \\
[\![\text{div2}]\!](a)(b) &= a \div 2^b
\end{aligned}$$

It is now a "hard fact of arithmetic" that our semantics is appropriate.

$$\dfrac{\dfrac{div2 : r \to (-r \to r) \quad 3 : r}{div2\ 3 : -r \to r} \qquad \dfrac{\dfrac{minus : -r \to (r \to -r) \quad 1 : -r}{minus\ 1 : r \to -r} \qquad 3 : r}{minus\ 1\ 3 : -r}}{div2\ 3\ minus\ 1\ 3 : r}$$

$$[\![div2\ 3\ minus\ 1\ 3 :\ r]\!] \quad = \quad 3 \div 2^{(1-3)} \quad = \quad 3 \div \frac{1}{4} \quad = \quad 12.$$

We don't need all this machinery to do this!

Let's introduce "variables" of type r.

$$\dfrac{\dfrac{div2 : r \to (-r \to r) \quad 3 : r}{div2\ 3 : -r \to r} \qquad \dfrac{\dfrac{minus : -r \to (r \to -r) \quad 1 : -r}{minus\ 1 : r \to -r} \qquad x : r}{minus\ 1\ x : -r}}{div2\ 3\ minus\ 1\ x : r}$$

$$[\![div2\ 3\ minus\ 1\ x]\!] \quad = \quad \text{the function taking } a \text{ to } 3/2^{(1-a)}.$$

The point now is that we can see from the type $x : r$
that this function is monotone (increasing) in x

(If the type were $x : -r$, it would be antitone in this occurrence.)

$$\dfrac{\dfrac{div2 : r \to (-r \to r) \quad 3 : r}{div2\ 3 : -r \to r} \qquad \dfrac{\dfrac{minus : -r \to (r \to -r) \quad 1 : -r}{minus\ 1 : r \to -r} \qquad x : r}{minus\ 1\ x : -r}}{div2\ 3\ minus\ 1\ x : r}$$

$$[\![x]\!] : R \to R$$

is monotone.

$$[\![minus\ 1\ x]\!] : R \to -R$$

is monotone,
since it is the function taking a to $[\![minus\ 1]\!](a)$,
and $[\![minus\ 1]\!] : R \to -R$.

$$\dfrac{\dfrac{div2 : r \to (-r \to r) \quad 3 : r}{div2\ 3 : -r \to r} \qquad \dfrac{\dfrac{minus : -r \to (r \to -r) \quad 1 : -r}{minus\ 1 : r \to -r} \qquad x : r}{minus\ 1\ x : -r}}{div2\ 3\ minus\ 1\ x : r}$$

$$[\![div2\ 3\ minus\ 1\ x\colon\ r]\!] : R \to R$$

is monotone,
since it is the composition

$$[\![div2\ 3\ x]\!] \circ [\![minus\ 1\ x\colon\ -r]\!],$$

and the composition of two anitone functions is monotone.

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 $\sigma \to \tau$.

To define the syntax of the calculus, we start with a set of *typed constants* $\mathsf{c} : \sigma$,
and also a set of *typed variables* $\mathsf{x} : \sigma$.

Definition

We define typed terms in the following way:

$$\frac{}{\mathsf{c} : \sigma} \qquad \frac{\mathsf{f} : \sigma \to \tau \quad \mathsf{t} : \sigma}{\mathsf{f}(\mathsf{t}) : \tau} \qquad \frac{\mathsf{x} : \sigma \quad \mathsf{t} : \tau}{\lambda x.t : \sigma \to \tau}$$

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

We return to the linguistic example, using base types e and t.
We abbreviate $e \dot{\to} t$ by p (for "property").

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

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

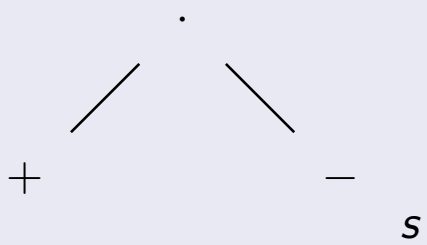
This is the most specific type we could assign to every.
But we could also say

$$\text{every} : p \xrightarrow{-} (p \dot{\to} t),$$

or even

$$\text{every} : p \dot{\to} (p \dot{\to} t).$$

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

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

Review of the semantics of the typed lambda calculus

Definition

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

For the base types $\beta \in \mathcal{B}$ there is no requirement on X_β.

For complex types $\sigma \to \tau$,
$X_{\sigma \to \tau}$ is the set of all functions from X_σ to X_τ.

The semantics begins with interpretations of the constants, requiring that $[\![\mathsf{c}]\!] \in X_\sigma$ whenever $\mathsf{c} : \sigma$.

The rest of the semantics is fairly natural, and it need not concern us here.

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,

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

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 $\mathcal{S} \models s \leq t$.

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 $s(v)[u^{m \cdot l}]$.

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 $[v^{\uparrow}]$ $[w^{\uparrow}]$.

Suppose $t[s^l]$ is a term with subterm s labeled by $l \in \{+, -\}$.

Let $\mathcal{S}$ be a standard structure.
Assume that $[\![s]\!]_{\hat{\sigma}} \leq_{\hat{\sigma}} [\![s']\!]_{\hat{\sigma}}$.

Then

1. If $l = +$, then $[\![t]\!]_{\hat{\tau}} \leq_{\hat{\tau}} [\![t^{s' \leftarrow s}]\!]_{\hat{\tau}}$
2. If $l = -$, then $[\![t^{s' \leftarrow s}]\!]_{\hat{\tau}} \leq_{\hat{\tau}} [\![t]\!]_{\hat{\tau}}$.

The Monotonicity Calculus

I have omitted the types

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

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

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

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

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

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

Soundness/Completeness Theorem

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

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

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

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

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

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

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.

This restriction leads to two extra rules in the calculus, unfortunately.

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

Soundness/Completeness Theorem

The following are equivalent:

- $\Gamma \models t \leq u$ on all models where the base types are interpreted as weakly complete orders.
- $\Gamma \vdash t \leq u$ in the proof system below.

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

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

$$\frac{u \leq v}{f^{\uparrow}(u) \leq f^{\uparrow}(v)}\ (\text{Monotone}) \qquad \frac{u \leq v}{f^{\downarrow}(v) \leq f^{\downarrow}(u)}\ (\text{Antitone})$$

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

Some key derivations for the completeness proof

Suppose that $f : \sigma \xrightarrow{+} \tau$, $g : \sigma \xrightarrow{-} \tau$, and $a, b : \sigma$.
If

$$\Gamma = \{f \leq g, a \leq b\} \text{ or } \Gamma = \{f \leq g, b \leq a\},$$

then $\Gamma \vdash f(a) \leq g(b)$.

$$\dfrac{\dfrac{a \leq b}{f(a) \leq f(b)}\,(\textsc{Mono}) \quad \dfrac{f \leq g}{f(b) \leq g(b)}\,(\textsc{Point})}{f(a) \leq g(b)}\,(\textsc{Trans})$$

$$\dfrac{\dfrac{f \leq g}{f(a) \leq g(a)}\,(\textsc{Point}) \quad \dfrac{b \leq a}{g(a) \leq g(b)}\,(\textsc{Anti})}{f(a) \leq g(b)}\,(\textsc{Trans})$$

Outline of the completeness proof

1. Γ is a set of inequalities over some vocabulary $\mathcal{V} = (\mathsf{Con}, \mathcal{T})$. Let $\mathcal{V}^+$ be a rich extension of $\mathcal{V}$. We consider Γ as a set of inequalities over $\mathcal{V}^+$. We also consider the preorders $\mathcal{C}_\sigma$ given by provability from Γ. The elements of each $\mathcal{C}_\sigma$ is the set of terms $t : \sigma' \preceq \sigma$, where t is a term over $\mathcal{V}^+$.
2. We find preorders $\mathbb{D}_\beta$ for all base types β. Unlike $\mathcal{C}_\beta$, $\mathbb{D}_\beta$ will be a CPL.
3. We automatically obtain the semantic domains $\mathbb{D}_\sigma$ when σ is a function type. We use our work in Section **??** to define maps $k_\sigma : \mathcal{C}_\sigma \to \mathbb{D}_\sigma$. Each k_σ is a one-to-one order embedding: it preserves and reflects equality and order.
4. We then define interpretations $[\![c{:}\sigma]\!] \in \mathbb{D}_\sigma$ *for all constants* c*, when* $\sigma = \mathcal{T}(c)$*. This gives a model* $\mathcal{M}$*,*

Why is the completeness work so hard?

The most natural strategy is to assume that

$$\Gamma \nvdash s \leq t$$

and to build a model of Γ where the inequality fails.

And the time-honored strategy is to build a model from the syntax.

This is hard here, because the orderings are so complicated.

Another option is to go for weaker results, via Henkin-type models.

This is the way things go in tomorrow's lecture,
where we still lack a "real" completeness result.

Tomorrow: next steps in this line of work

We are going 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.

The Monotonicity Calculus vs. the equational logic of function application

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

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

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

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

$$\frac{t = u}{u = t}\ (\textsc{Symmetric}) \qquad \frac{f = g \quad t = u}{f(t) = g(u)}\ (\textsc{Application})$$