Monotonicity and Polarity in Natural Logic

Larry Moss, Indiana University

Workshop on Semantics for Textual Inference, July 10, 2011
“Natural Logic is a cover term for a family of formal approaches to semantics and textual inferencing as currently practiced by computational linguists.

“They have in common a proof theoretical rather than a model-theoretic focus and an overriding concern with feasibility.”

_Natural Logic_ sometimes refers just to work on monotonicity, but in this talk I’ll be broader.
Review

First-order logic

$FO^2 + \text{"} R \text{ is trans"}$

2 variable FO logic

$\dagger$ adds full $N$-negation

$RC(tr) +$ opposites

$RC +$ (transitive) comparative adjs

$R +$ relative clauses

$S +$ full $N$-negation

$R = \text{relational syllogistic}$

$S \geq$ adds $|p| \geq |q|$

$S$: all/some/no $p$ are $q$
Complexity

(MOSTLY) BEST POSSIBLE RESULTS ON THE VALIDITY PROBLEM

- Undecidable
  - Church 1936
  - Grädel, Otto, Rosen 1999

- In co-NEXPTIME
  - EXPTIME
  - Lutz & Sattler 2001

- Co-NEXPTIME
  - Grädel, Kolaitis, Vardi '97
  - EXPTIME
  - Pratt-Hartmann 2004

- Lower bounds also open

- Co-NP
  - McAllester & Givan 1992

- NLOGSPACE
What are the simplest forms of reasoning?

- Monotonicity in both mathematics and language
- Equational reasoning
- Syllogistic reasoning
Example of mathematical reasoning with monotone and antitone functions

Which is bigger?

\[
\left( 7 + \frac{1}{4} \right)^{-3} \quad \text{or} \quad \left( 7 + \frac{1}{\pi^2} \right)^{-3}
\]
Which is bigger?

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

\[
\frac{2 \leq \pi}{\frac{1}{\pi} \leq \frac{1}{2}} \quad 1/x \text{ is antitone}
\]

\[
\frac{\frac{1}{\pi} \leq \frac{1}{2}}{\frac{1}{\pi^2} \leq \frac{1}{4}} \quad x^2 \text{ is monotone}
\]

\[
\frac{\frac{7 + \frac{1}{\pi^2} \leq 7 + \frac{1}{4}}{7 + \frac{1}{\pi^2} \leq 7 + \frac{1}{4}}} \quad 7 + x \text{ is monotone}
\]

\[
\frac{(7 + \frac{1}{4})^{-3} \leq (7 + \frac{1}{\pi^2})^{-3}}{\frac{(7 + \frac{1}{4})^{-3} \leq (7 + \frac{1}{\pi^2})^{-3}}{x^{-3} \text{ is antitone}}}
\]
A first monotonicity judgment for language

Assume: $\text{barks loudly} \leq \text{barks} \leq \text{vociferates}$

Notice that if we replace $\text{barks}$ by a “bigger” word, we have an inference.

For example:

\[
\begin{array}{c}
\text{every dog barks} \\
\hline
\text{every dog vociferates}
\end{array}
\]
A first monotonicity judgment for language

*every dog barks*

Assume: \( \text{barks loudly} \leq \text{barks} \leq \text{vociferates} \)
Notice that if we replace \( \text{barks} \) by a “bigger” word, we have an inference.
For example:

\[
\begin{align*}
\text{every dog barks} \\
\hline
\text{every dog vociferates}
\end{align*}
\]

**Notation**

We’ll indicate this by

\( \text{every dog barks} \uparrow \)
Assume: \( \text{barks loudly} \leq \text{barks} \leq \text{vociferates} \)

Assume: \( \text{old dog} \leq \text{dog} \leq \text{animal} \)

We want

\[
\begin{align*}
\text{every dog} & \downarrow \text{barks} \uparrow \\
\text{no dog} & \downarrow \text{barks} \\
\text{not every dog} & \uparrow \text{barks} \\
\text{some dog} & \uparrow \text{barks} \\
\text{most dogs} & \times \text{bark} \uparrow
\end{align*}
\]

no monotonicity in first argument
**A categorial lexicon**

| (Dana, $NP$)     |
| (Kim, $NP$)      |
| (smiled, $NP/\text{S}$) |
| (laughed, $NP/\text{S}$) |
| (cried, $NP/\text{S}$) |
| (praised, $(NP/\text{S})/NP$) |
| (teased, $(NP/\text{S})/NP$) |
| (interviewed, $(NP/\text{S})/NP$) |
| (joyfully, $(NP/\text{S})\backslash(NP/\text{S})$) |
| (carefully, $(NP/\text{S})\backslash(NP/\text{S})$) |
| (excitedly, $(NP/\text{S})\backslash(NP/\text{S})$) |
A parse tree showing that Dana smiled joyfully is an $S$.
The semantics of CG

It works by

► Assigning sets to the base types, here $NP$, $S$.
► Using function sets for the slash types
► Giving fixed meanings to the lexical items
► Working up the tree using function application

The previous stuff gives a model.

Overall semantic facts are defined in terms of models, as we have already seen.
For this talk, simpler base types will do

$pr$ for “property”, $t$ for “truth value”.

Also, I’ll ignore the directionality of the slash arrows to make things much simpler, and to highlight what is new here.

\begin{align*}
\text{every} & : (pr, (pr, t)) \\
\text{some} & : (pr, (pr, t)) \\
\text{no} & : (pr, (pr, t)) \\
\text{any} & : (pr, (pr, t))
\end{align*}

(Note that we already have a problem in giving the semantics of “any”.)
A preorder is a pair $\mathbb{P} = (P, \leq)$, where $\leq$ is reflexive and transitive.

Preorders are needed to really discuss upward/downward monotonicity.

The proposal is to enrich the basic semantic architecture of CG by moving from sets to preorders.
A preorder is a pair $\mathcal{P} = (P, \leq)$, where $\leq$ is reflexive and transitive.

Preorders are needed to really discuss upward/downward monotonicity.

The proposal is to enrich the basic semantic architecture of CG by moving from sets to preorders.

A function $f : \mathcal{P} \to \mathcal{Q}$ is

- monotone if $p \leq q$ in $\mathcal{P}$ implies $f(p) \leq f(q)$ in $\mathcal{Q}$.
- antitone if $p \leq q$ in $\mathcal{P}$ implies $f(q) \leq f(p)$ in $\mathcal{Q}$. 
A preorder is a pair \( \mathbb{P} = (P, \leq) \), where \( \leq \) is reflexive and transitive.

Preorders are needed to really discuss upward/downward monotonicity.

The proposal is to enrich the basic semantic architecture of CG by moving from sets to preorders.

**A function** \( f : \mathbb{P} \rightarrow \mathbb{Q} \) is

- monotone if \( p \leq q \) in \( \mathbb{P} \) implies \( f(p) \leq f(q) \) in \( \mathbb{Q} \).
- antitone if \( p \leq q \) in \( \mathbb{P} \) implies \( f(q) \leq f(p) \) in \( \mathbb{Q} \).

**From now on, all functions are monotone**

- \( \mathbb{Q} \) is \( (Q, \geq) \): it’s \( Q \) upside-down.

- \( -Q \) is \( (Q, \geq) \): it’s \( Q \) upside-down.

- \( -(\mathbb{Q}) = \mathbb{Q} \).

An antitone \( f : \mathbb{P} \rightarrow \mathbb{Q} \) is exactly a montone \( f : \mathbb{P} \rightarrow -\mathbb{Q} \).
Let’s think about monotonicity in connection with truth tables

*T* means “true” and *F* means “false”.

\[ \neg P: \text{not } P \]
\[ P \land Q: P \text{ and } Q. \]
\[ P \lor Q: P \text{ or } Q. \]
\[ P \rightarrow Q: P \text{ implies } Q; \text{ or If } P, \text{ then } Q. \]

<table>
<thead>
<tr>
<th></th>
<th>(-P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(P)</th>
<th>(-P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(P)</th>
<th>(Q)</th>
<th>(P \land Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(P)</th>
<th>(Q)</th>
<th>(P \lor Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(P)</th>
<th>(Q)</th>
<th>(P \rightarrow Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
But what are the preorders?

The main preorder here is the tiny preorder I’ll call $\mathbb{2}$.

Notice that $F < T$. 
But what are the preorders?

But for $\land$, $\lor$, and $\rightarrow$, we need to think about pairs of truth values, so we need a preorder with four elements.

Which should we use?
But what are the preorders?
Conjunction $\land$ as a monotone function

\begin{align*}
(T, T) & \\
(T, F) & \\
(F, F) & \\
(F, T) &
\end{align*}

$2 \times 2$
Disjunction $\lor$ as a monotone function

\[
\begin{array}{c}
(T, T) \\
(T, F) \\
(F, F) \\
(F, T)
\end{array}
\to
\begin{array}{c}
T \\
F
\end{array}
\]

$2 \times 2 \to 2$
Is it a monotone function from $2 \times 2$ to $2$?
Is negation monotone?
The opposite of an order

2

2^{op}
Negation is antitone

This is the same as a monotone function from $-2$ to $2$
**Negation is antitone**

This is the same as a monotone function from $-2$ to $2$
Let's go back to implication →

\[
\begin{array}{c}
(T, T) \\
(T, F) \\
(F, F) \\
(F, T)
\end{array}
\]

\[
\begin{array}{c}
T \\
F
\end{array}
\]

\[
2 \times 2
\]

\[
2
\]
Find a preorder $\mathbb{P}$ so that $\rightarrow$ is a monotone function from $\mathbb{P}$ to $\mathcal{P}$.

Hint: it’s not $(\mathcal{P} \times -\mathcal{P})$, but this is on the right track.
Find a preorder $\mathcal{P}$ so that $\rightarrow$ is a monotone function from $\mathcal{P}$ to $\mathcal{2}$.

Hint: try the orders below:

- $\mathcal{P} = \{ (F, T), (F, F), (T, F), (T, T) \}$
- $\mathcal{P} = \{ (T, F), (F, F), (F, T), (T, T) \}$

$\mathcal{2} \times \mathcal{2}$ $\times \mathcal{2} \times \mathcal{2}$
Now we can settle the matter about implication $\rightarrow$

It is a monotone function from $-2 \times 2$ to 2
Let \([P, Q]\) be the monotone function preorder.

\[ [P, -Q] = -[-P, Q] \]

This means that any lexical items typed as \(P \rightarrow -Q\)
could just as well be typed as \(-P \rightarrow Q\).

However, the orders \([P, -Q]\) and \([-P, Q]\) are opposites.
Take *categorial grammar* a la Ajdukiewicz-Bar Hillel-Lambek and interpret the syntactic types not in sets but in preorders, adding the ability to use *opposite* of a preorder as well.

**Examples of typed constants:**

\[
\begin{align*}
ev_{\rightarrow}^+ & : (-pr, (pr, t)) & ev_{\rightarrow}^- & : (pr, (-pr, -t)) \\
s_{\rightarrow}^+ & : (pr, (pr, t)) & s_{\rightarrow}^- & : (-pr, (-pr, -t)) \\
n_{\rightarrow}^- & : (-pr, (-pr, t)) & n_{\rightarrow}^- & : (pr, (pr, t)) \\
an_{\rightarrow}^+ & : (-pr, (pr, t)) & an_{\rightarrow}^- & : (-pr, (-pr, -t)) \\
\end{align*}
\]

Every binary atom \( r \) gives four type constants:

\[
\begin{align*}
r_1^+ & : ((pr, t), pr) & r_1^- & : ((-pr, t), -pr) \\
r_2^+ & : ((-pr, t), pr) & r_2^- & : ((pr, t), -pr) \\
\end{align*}
\]
We begin with a set \( \mathcal{T}_0 \) of basic types: for simplicity \( pr \) and \( t \). We then form a set \( \mathcal{T}_1 \) of types as follows:

\[
\mathcal{T}_0 \subseteq \mathcal{T}_1.
\]

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 \( (\sigma, \tau) \equiv (\sigma', \tau') \).

**The set of types**

\[
\mathcal{T} = \mathcal{T}_1/\equiv.
\]
Proposal: use preorders for the semantic spaces

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

$$
\mathbb{P}(\sigma, \tau) = [\mathbb{P}_\sigma, \mathbb{P}_\tau] \quad \text{monotone function space}
$$

$$
\mathbb{P}_{-\sigma} = -\mathbb{P}_\sigma \quad \text{opposite preorder}
$$

If $\sigma \equiv \tau$, then $\mathbb{P}_\sigma = \mathbb{P}_\tau$. So we have $\mathbb{P}_\sigma$ for $\sigma \in \mathcal{T}$. We use $P_\sigma$ 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 $f(v) \in P_\sigma$. 
Some semantic interpretations in a universe $X$

$\emptyset$ is true $<$ false. $X$ is the flat preorder on a set $X$

$[X, 2]$ is in one-to-one correspondence with the set of subsets of $X$.

- every $\in [\neg [X, 2], [[X, 2], 2]]$
- some $\in [[X, 2], [[X, 2], 2]]$
- no $\in [\neg [X, 2], [\neg [X, 2], 2]]$

in the standard way:

$$\text{every}(p)(q) = \begin{cases} 
\text{true} & \text{if } p \leq q \\
\text{false} & \text{otherwise}
\end{cases}$$

$$\text{some}(p)(q) = \neg \text{every}(p)(\neg \circ q)$$

$$\text{no}(p)(q) = \neg \text{some}(p)(q)$$

It follows from the Main Fact above that

- every $\in [[X, 2], [\neg [X, 2], \neg 2]]$
- some $\in [\neg [X, 2], [\neg [X, 2], \neg 2]]$
- no $\in [[X, 2], [[X, 2], \neg 2]]$
Examples

\[
\begin{align*}
\text{every}^- : (pr, (−pr, −t)) & \quad \text{cat}^+ : pr \\
\text{chase}_1^- : ((−pr, −t), −pr) & \quad \text{every}^-(\text{cat}^+) : (−pr, −t) \\
\text{chase}_1^- (\text{every}^-(\text{cat}^+)) : −pr
\end{align*}
\]

\[
\begin{align*}
\text{some}^+(\text{dog}^+)(\text{chase}_1^+(\text{every}^+(\text{cat}^-))) : t \\
\text{some}^+(\text{dog}^+)(\text{chase}_2^+(\text{no}^+(\text{cat}^-))) : t \\
\text{no}^+(\text{dog}^-)(\text{chase}_2^-(\text{no}^+(\text{cat}^+))) : t
\end{align*}
\]

Theorem

The +, − signs automatically indicate the polarity.
Another

\[
\begin{align*}
\text{any}^- &: (-pr, (-pr, -t)) \quad \text{cat}^- : -pr \\
\text{see}_2^- &: ((-pr, -t), -pr) \quad \text{any}^-(\text{cat}^-) : (-pr, -t) \\
\text{every}^+ &: (-pr, (pr, t)) \quad \text{see}_2^- (\text{any}^-(\text{cat}^-)) : -pr \\
\text{every}^+ (\text{see}_2^- (\text{any}^-(\text{cat}^-))) &: (pr, t) \\
\text{runs}^+ &: \\
\text{every}^+ (\text{see}_2^- (\text{any}^-(\text{cat}^-))(\text{runs}^+)) &: t
\end{align*}
\]

Note that any$^+$ and any$^-$ should not have the same interpretation!!

\[
\begin{align*}
\text{any}^- &= \text{some} \quad \text{any}^+ = \text{every}
\end{align*}
\]

Compare

\[
\text{any}^+ (\text{cat}^-)(\text{see}_1^- (\text{any}^+(\text{dog}^-))) : t.
\]
But it’s open to get completeness for this logic, and in fact there are interesting questions:

\[
\begin{align*}
\text{every}^+(\text{see}_1^- (\text{every}^- (\text{cat}^+))) (\text{see}_1^+ (\text{every}^+ (\text{cat}^-))) \\
\text{every}^+(\text{see}_1^- (\text{any}^- (\text{cat}^+))) (\text{see}_1^+ (\text{any}^+ (\text{cat}^-)))
\end{align*}
\]

See also Zamansky, Francez and Winter, 2006.
For me:

- It would be a step towards a complete logic for a significant language

For those in RTE:

- The sound principles give transformation rules.
- Completeness would be secondary.
- Logical systems are often implemented, and then this could be useful.