The Logic of Sense and Reference

Reinhard Muskens

Tilburg Center for Logic and Philosophy of Science (TiLPS)

ESSLLI 2009, Day 3

Overview

1. Sense and the Computation of Reference
2. Circular Propositions
3. Conclusion

Senses as Algorithms

- Moschovakis (1994) gives an interesting explication of Frege's distinction between sense and reference: senses are algorithms and references are values.
- This view agrees well with Frege's (1892) explanation of sense as the Art des Gegebenseins of a referent (the way the referent is given, or 'mode of presentation').
- A relatively recent application of Moschovakis' ideas to linguistics can be found in van Lambalgen and Hamm (2003).
- The idea throws light on two foundational problems in semantics: the problem of intensionality and Liar-like phenomena.

Hyperintensionality

- Consider the following sentences:
  (a) The cat is out if the dog is in
  (b) The dog is out if the cat is in
  (c) Fritz is aware that the cat is out if the dog is in
  (d) Fritz is aware that the dog is out if the cat is in
- (a) and (b) co-entail, but (c) and (d) do not.
- This seems problematic when one is modeling natural language with the help of logic. In standard logics one has replacement laws like $\varphi \leftrightarrow \psi \models [\varphi/p]\chi \leftrightarrow [\psi/p]\chi$
- The crucial point seems to be that 'propositional attitudes' like aware do not express a relation between a person and the truth conditions of a sentence, but between persons and 'senses'. Identity of senses implies identity of truth conditions, but not vice versa.

The Liar and Friends

- This sentence is false (Liar)
- This sentence is true (Truth-teller)
- (a) Sentence (b) is false
  (b) Sentence (a) is true (Liar cycle)
- Most of Nixon's assertions about Watergate are false
  (could be a contingent Liar, see Kripke 1975)

The Senses-as-Algorithms Idea

- Two different algorithms can have the same input-output behaviour. This is related to the hyperintensionality problem.
- Algorithms are by no means guaranteed to halt. This is related to the Liar.
- There are, I think, connections to the computational theory of mind which holds that the mind is a large collection of algorithms for all kinds of tasks: vision, face recognition, reasoning, language, etc.
- (This last point is a very unFregean thought.)

Needed: a Simple Formalization

- Moschovakis formalizes his senses-as-algorithms idea with the help of a formal system that is obtained by adding recursion to first order logic. The result is heavy artillery.
- But the result is also first-order, while for the treatment of natural language we would like to have a type logic.
- Moschovakis (2006) gives a higher-order Montague-like system, but this more recent system does not treat Liar-like phenomena (no self-reference).
- My purpose here is to sketch a simple logical system that is consistent with the view that propositions are algorithms and is also consistent with many of the insights about natural language semantics that have arisen in the Montague tradition (broadly conceived).
- See Muskens (2005) for technical details.

Using Thomason's Intentional Logic

- Last Monday I presented a streamlined version of Thomason's (1980) theory of Intentional Logic.
- Key to Thomason's theory is that he uses primitive propositions and then uses some function to send these to some conventional kind of semantic objects (such as sets of possible worlds).
- The following few slides are taken from Monday's lecture and are meant as a recap.
- But this time we will use Thomason's ideas as a starting point for a theory of meanings as algorithms.
- It will be argued that, with the right meaning postulates (axioms) in place, the primitive type p objects start to behave like algorithms.

A Set of Non-logical Constants

Non-logical Constants	Type
not	pp
and, or, if	$p(pp)$
every, a, no, the	$(ep)((ep)p)$
is, love, kiss, ...	$e(ep)$
hesperus, phosphorus, mary, ...	$(ep)p$
planet, man, woman, run, ...	ep
necessarily, possibly	pp
believe, know, aware	$p(ep)$
hesperus, phosphorus, mary, ...	e
love, kiss, ...	$e(e(st))$
planet, man, woman, ...	$e(st)$
acc	$s(st)$
believe, know, aware	$p(e(st))$

Some Terms of Type p

1. ((a woman)walk)
2. ((no man)talk)
3. (hesperus λx((a planet)(is x)))
4. ((if((a woman)walk))((no man)talk))
5. ((if((a man)talk))((no woman)walk))
6. (mary(aware((if((a woman)walk))((no man)talk))))
7. (mary(aware((if((a man)talk))((no woman)walk))))
8. ((a woman)λx(mary(aware((if(walk x))((no man)talk)))))

From Propositions to Sets of Worlds I

- At this point in Monday's lecture, a function r was given sending primitive propositions to sets of worlds.
- In practice, application of the function r to some term A of type p would lead to a computation recurring on the structure of A.
- We will now give a more explicitly computational picture of the relation between type p objects and sets of possible worlds.
- This time we will use a relation d to make the connection. Read $d(\pi, \tau)$ as 'proposition π determines the set of worlds τ'.
- Functionality of d may (or may not) be required: $\forall p \forall \tau \tau' [[d(p, \tau) \wedge d(p, \tau')] \rightarrow \tau = \tau']$
- But d need not be total: computations may diverge.

From Propositions to Sets of Worlds II

- Meaning postulates such as the following can be adopted:
  $d(\pi, \tau) \rightarrow d(\mathsf{not}\ \pi, \lambda i.\neg \tau i)$
  $d(\pi, \tau) \wedge d(\pi', \tau') \rightarrow d(\mathsf{and}\ \pi\pi', \lambda i.\tau i \wedge \tau' i)$
- These have a declarative meaning, but are also close to clauses in a logic program.
- Procedural meaning: In order to find a τ' such that $d(\mathsf{not}\ \pi, \tau')$, find a τ such that $d(\pi, \tau)$ and unify τ' with $\lambda i.\neg \tau i$.

A Generalization

In fact, we will need a generalization of the predicate d. While d relates objects of type p with objects of type st, we need relations d^k that connect objects of type $e^k p$ with those of type $e^k(st)$. We will continue to write d^0 as d.

Meaning Postulates

There is a meaning postulate for each of the constants in sans serif non-italic. Here are some examples:

- $d^n(\varrho, R) \wedge d^n(\varrho', R') \rightarrow d^n(\lambda\vec{z}.\mathsf{and}(\varrho\vec{z})(\varrho'\vec{z}), \lambda\vec{z}\lambda i.R\vec{z}i \wedge R'\vec{z}i)$
- $d^{n+1}(\varrho, R) \wedge d^{n+1}(\varrho', R') \rightarrow$
  $d^n(\lambda\vec{z}.\mathsf{every}(\varrho'\vec{z})(\varrho\vec{z}), \lambda\vec{z}\lambda i\forall x[R'\vec{z}xi \rightarrow R\vec{z}xi])$
- $d^n(\varrho, R) \rightarrow d^n(\lambda\vec{z}.\mathsf{necessarily}(\varrho\vec{z}), \lambda\vec{z}\lambda i.\forall j[acc\, ij \rightarrow R\vec{z}j])$
- $d^{n+2}(\lambda\vec{u}.\mathsf{love}\ xy, \lambda\vec{u}.love\, xy)$, where $\vec{u}$ contains x and y
- $d^{n+1}(\lambda\vec{z}.\mathsf{believe}\,(\varrho\vec{z}), \lambda\vec{z}.believe\,(\varrho\vec{z}))$

Together the meaning postulates form a logic program.

A Refutation

$$\leftarrow \underline{d((\text{if}((\text{a woman})\text{walk}))((\text{no man})\text{talk}), \tau)}$$
$$\downarrow \tau := \lambda i.\tau_1 i \rightarrow \tau_2 i$$
$$\leftarrow d(((\text{a woman})\text{walk}), \tau_1),\ \underline{d(((\text{no man})\text{talk}), \tau_2)}$$
$$\downarrow \tau_2 := \lambda i.\neg\exists x[P_1 x i \wedge P_2 x i]$$
$$\leftarrow d(((\text{a woman})\text{walk}), \tau_1),\ \underline{d^1(\text{man}, P_1)},\ d^1(\text{talk}, P_2)$$
$$\downarrow P_1 := man$$
$$\leftarrow \underline{d(((\text{a woman})\text{walk}), \tau_1)},\ d^1(\text{talk}, P_2)$$
$$\downarrow \tau_1 := \lambda i.\exists y[P_3 y i \wedge P_4 y i]$$
$$\leftarrow \underline{d^1(\text{woman}, P_3)},\ d^1(\text{walk}, P_4),\ d^1(\text{talk}, P_2)$$
$$\downarrow P_3 := woman$$
$$\leftarrow d^1(\text{walk}, P_4),\ \underline{d^1(\text{talk}, P_2)}$$
$$\downarrow P_2 := talk$$
$$\leftarrow \underline{d^1(\text{walk}, P_4)}$$
$$\downarrow P_4 := walk$$
$$\leftarrow$$

A Refutation (continued)

- Composition of substitutions gives
  $\tau = \lambda i.\exists x[woman\, xi \wedge walk\, xi] \rightarrow \neg\exists x[man\, xi \wedge talk\, xi]$.
- Used higher order patterns unification (Miller 1991).
  - a pattern is a term M such that for every subterm of M of the form $X\, M_1 \ldots M_n$, where X is a free variable, the terms $M_1, \ldots, M_n$ are distinct variables bound in M.
  - terms $d^k(M, M')$ in meaning postulates are all of this form.
  - decidable in polynomial time +
  - when a unification problem has a unifier it has a most general unifier (Miller 1991).
- Patterns unification probably still is a huge overkill for this particular problem.

Another Refutation

$$\leftarrow d((\text{a man})(\lambda x.\text{necessarily}((\text{every unicorn})(\lambda y.\text{kiss } yx))), \tau)$$

$$\downarrow \tau := \lambda i \exists x[P_1 xi \wedge P_2 xi]$$

$$\leftarrow d^1(\text{man}, P_1),\ d^1(\lambda x.\text{necessarily}((\text{every unicorn})(\lambda y.\text{kiss } yx)), P_2)$$

$$\downarrow P_1 := man$$

$$\leftarrow d^1(\lambda x.\text{necessarily}((\text{every unicorn})(\lambda y.\text{kiss } yx)), P_2)$$

$$\downarrow P_2 := \lambda x \lambda i \forall j[acc\ ij \rightarrow P_3 xj]$$

$$\leftarrow d^1(\lambda x.(\text{every unicorn})(\lambda y.\text{kiss } yx), P_3)$$

$$\downarrow P_3 := \lambda x \lambda i \forall y[R_1 xyi \rightarrow R_2 xyi]$$

$$\leftarrow d^2(\lambda x.\text{unicorn}, R_1),\ d^2(\lambda xy.\text{kiss } yx, R_2)$$

$$\downarrow R_1 := \lambda x.unicorn$$

$$\leftarrow d^2(\lambda xy.\text{kiss } yx, R_2)$$

$$\downarrow R_2 := \lambda xy.kiss\ yx$$

$$\leftarrow$$

$$\tau = \lambda i \exists x[man\ xi \wedge \forall j[acc\ ij \rightarrow \forall y[unicorn\ yj \rightarrow kiss\ yxj]]]$$

Hyperintensionality

Define entailment between $\mathcal{T}^p_{\mathrm{LF}}$ terms on the basis of the type st terms associated with them by d.

1. ((if((a woman)walk))((no man)talk))
2. $\lambda i.\exists x[woman\ xi \wedge walk\ xi] \rightarrow \neg\exists x[man\ xi \wedge talk\ xi]$
3. ((if((a man)talk))((no woman)walk))
4. $\lambda i.\exists x[man\ xi \wedge talk\ xi] \rightarrow \neg\exists x[woman\ xi \wedge walk\ xi]$
5. (mary(aware((if((a woman)walk))((no man)talk))))
6. $aware$((if((a woman)walk))((no man)talk))$\ mary$
7. (mary(aware((if((a man)talk))((no woman)walk))))
8. $aware$((if((a man)talk))((no woman)walk))$\ mary$

1 and 3 co-entail but may be different objects; therefore 5 and 7 do not co-entail.

Propositions as Algorithms/Queries

We should distinguish between

- A type $\mathcal{T}^{p}_{\mathrm{LF}}$ term S,
- whatever it denotes (a sense), and
- the query $\leftarrow d(S, X)$.

But considering that the role of a sense in this theory is essentially that of returning a referent (if there is one), we may start to think of a sense S as the query $\leftarrow d(S, X)$, or the query $\leftarrow d(S, X)$ plus the database of meaning postulates.

Circular Propositions

- From the viewpoint of linguistic semantics the Liar is important because it shows that the semantic system can get into trouble if very common elements are combined in a special way.
- It is to be *expected* from a computational device with a biological origin that computations may loop in unusual circumstances.
- Our task is not to 'solve' the paradox, but to give a formal account of normal practices and show how these can lead to trouble in special situations.

Ingredients of the Liar

- We will treat the predicates true and false in an entirely trivial way:
  $d(\pi, \tau) \rightarrow d(\mathsf{true}\ \pi, \tau)$
  $d(\pi, \tau) \rightarrow d(\mathsf{false}\ \pi, \lambda i.\neg \tau i)$
- We need demonstratives 'this' and 'that' that can refer to propositions and a relation ant of type $p(pt)$ that holds between a demonstrative and its antecedent. The following seem reasonable.
  $ant(\mathsf{this}, \pi) \wedge d(\pi, \tau) \rightarrow d(\mathsf{this}, \tau)$
  $ant(\mathsf{that}, \pi) \wedge d(\pi, \tau) \rightarrow d(\mathsf{that}, \tau)$
- If a demonstrative is understood to have a certain referent, we add that fact to the database.

Normal Use of these Ingredients

- If some woman is walking no man is talking. That's true.
- Add to database:
  $ant(\mathsf{that}, (\mathsf{if}((\mathsf{a\ woman})\mathsf{walk}))((\mathsf{no\ man})\mathsf{talk}))$
- A refutation tree for $d(\mathsf{true\ that}, \tau)$:

$\leftarrow d(\mathsf{true\ that}, \tau)$

$\downarrow$

$\leftarrow d(\mathsf{that}, \tau)$

$\downarrow$

$\leftarrow ant(\mathsf{that}, \pi),\ d(\pi, \tau)$

$\downarrow\ \pi := (\mathsf{if}((\mathsf{a\ woman})\mathsf{walk}))((\mathsf{no\ man})\mathsf{talk})$

$\leftarrow d((\mathsf{if}((\mathsf{a\ woman})\mathsf{walk}))((\mathsf{no\ man})\mathsf{talk}), \tau)$

$\vdots$

(continued as before)

Liar and Truth-teller

Add $ant(\mathsf{this}, \mathsf{false\ this})$ to the database in case of the Liar; $ant(\mathsf{this}, \mathsf{true\ this})$ in case of the Truth-teller.

$$
\begin{array}{l}
\leftarrow d(\mathsf{false\ this}, \tau) \\
\qquad \downarrow\ \tau := \lambda i.\neg\tau_1 i \\
\leftarrow d(\mathsf{this}, \tau_1) \\
\qquad \downarrow \\
\leftarrow ant(\mathsf{this}, \pi),\ d(\pi, \tau_1) \\
\qquad \downarrow\ \pi := \mathsf{false\ this} \\
\leftarrow d(\mathsf{false\ this}, \tau_1) \\
\qquad \downarrow\ \tau_1 := \lambda i.\neg\tau_2 i \\
\leftarrow d(\mathsf{this}, \tau_2) \\
\qquad \vdots
\end{array}
$$

a.– Liar

$$
\begin{array}{l}
\leftarrow d(\mathsf{true\ this}, \tau) \\
\qquad \downarrow \\
\leftarrow d(\mathsf{this}, \tau) \\
\qquad \downarrow \\
\leftarrow ant(\mathsf{this}, \pi),\ d(\pi, \tau) \\
\qquad \downarrow\ \pi := \mathsf{true\ this} \\
\leftarrow d(\mathsf{true\ this}, \tau) \\
\qquad \downarrow \\
\leftarrow d(\mathsf{this}, \tau) \\
\qquad \vdots
\end{array}
$$

b.– Truth-teller

Conclusion

- The view that senses are algorithms offers an interesting perspective on the foundations of natural language semantics.
- The view has something to contribute in at least two directions:
  - hyperintensionality and identity criteria for senses
  - Liar-like paradoxes and circularity.
- The approach matches well with a Montague-like declarative treatment of natural language semantics.
- But matches equally well with a more procedural view on cognition. This is because of the dual character of logic, which is both declarative and procedural.

Conclusion (continued)

- Formalization is easy, even trivial, and does not require machinery that is not needed independently. Although we build upon the ideas of Moschovakis, we do not need his technical apparatus.
- In the treatment of the Liar and friends divergence is a result of rules that work perfectly reasonable in standard cases.

References I

Frege, G. (1892).
Über Sinn und Bedeutung.
In G. Patzig (Ed.), *Funktion, Begriff, Bedeutung. Fünf Logische Studien.* Göttingen: Vanden Hoeck.

Kripke, S. (1975).
Outline of a Theory of Truth.
Journal of Philosophy 72, 690–716.

van Lambalgen, M. and F. Hamm (2003).
Moschovakis' Notion of Meaning as Applied to Linguistics.
In M. Baaz and J. Krajicek (Eds.), *Logic Colloquium '01*, ASL Lecture Notes in Logic.

Miller, D. (1991).
A Logic Programming Language with Lambda-abstraction, Function Variables, and Simple Unification.
Journal of Logic and Computation 1, 497–536.

References II

Moschovakis, Y. (1994).
Sense and Denotation as Algorithm and Value.
In *Logic Colloquium '90 (Helsinki 1990)*, Volume 2 of *Lecture Notes in Logic*, pp. 210–249. Berlin: Springer.

Moschovakis, Y. (2006).
A Logical Calculus of Meaning and Synonymy.
Linguistics and Philosophy 29, 27–89.

Muskens, R. (2005).
Sense and the Computation of Reference.
Linguistics and Philosophy 28(4), 473–504.

Thomason, R. (1980).
A Model Theory for Propositional Attitudes.
Linguistics and Philosophy 4, 47–70.