Lambda Grammars and Hyperintensionality

Reinhard Muskens

Tilburg University

Lambda Calculus and Formal Grammar
NII, Tokyo, 18 February 2005

Outline

Senses as Algorithms

- Moschovakis (1990) gives an interesting explication of Frege's (1892) distinction between sense and reference: senses are algorithms and references are values.
- This view agrees well with Frege's explanation of sense as the Art des Gegebenseins of a referent (the way the referent is given, or 'mode of presentation').
- A 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 not out
  (b) The dog is out if the cat is not out
  (c) Fritz is aware that the cat is out if the dog is not out
  (d) Fritz is aware that the dog is out if the cat is not out
- (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 (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 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.
- (Neither Frege nor Moschovakis would buy the last point, I think.)

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 (2003) 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).

Thomason's Intentional Logic

- We will build upon the 'Intentional Logic' of Thomason (1980) and introduce a type p of primitive propositions.
- Unlike Thomason, we will work in a classical type logic with ground types e (for entities), s (possible worlds), p (propositions or senses), and t (truth-values).
- It will be argued that, with the right meaning postulates (axioms) in force, the 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)))))

$\mathcal{T}^p_{\mathsf{LF}}$

- Consider the closed terms of type p built from constants in non-italic sans serif and variables, using application and linear abstraction. Denote this set with $\mathcal{T}^p_{\mathsf{LF}}$.
- Since all the constants that are used in $\mathcal{T}^p_{\mathsf{LF}}$ are non-logical, there is really not much logic here. We can do $\beta\eta$-conversions and that is basically all.
- There is a close similarity between these terms and the usual LF trees. Compare e.g. the LF from Heim and Kratzer's textbook in (a) with the type p term (b).

  (a) $[_S[_{DP}$ every linguist$][1[_S$ John$[_{VP}$ offended $t_1]]]]$

  (b) ((every linguist)λx_1(john (offend x_1)))
- (chains vs. linearity)

From Propositions to Sets of Worlds

- We will connect propositions with sets of possible worlds using a relation d. Read $d(\pi, \tau)$ as 'proposition π determines the set of worlds τ'.
- Functionality of d may (or may not) be required: $\forall p \forall \tau \tau' [[d(p, \tau) \wedge d(p, \tau')] \rightarrow \tau = \tau']$
- Meaning postulates such as the following can be adopted:
  $d(\pi, \tau) \rightarrow d(\text{not } \pi, \lambda i. \neg \tau i)$
  $d(\pi, \tau) \wedge d(\pi', \tau') \rightarrow d(\text{and } \pi \pi', \lambda i. \tau i \wedge \tau' i)$
- 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 $d(\text{not } \pi, \tau')$, 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 $XM_1 \ldots M_n$, 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. $\lambda i.\exists x[\mathit{woman}\ xi \wedge \mathit{walk}\ xi] \rightarrow \neg\exists x[\mathit{man}\ xi \wedge \mathit{talk}\ xi]$
3. ((if((a man)talk))((no woman)walk))
4. $\lambda i.\exists x[\mathit{man}\ xi \wedge \mathit{talk}\ xi] \rightarrow \neg\exists x[\mathit{woman}\ xi \wedge \mathit{walk}\ xi]$
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:
  $d(\pi, \tau) \rightarrow d(\text{true } \pi, \tau)$
  $d(\pi, \tau) \rightarrow d(\text{false } \pi, \lambda i.\neg\tau i)$
- 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.
  $ant(\text{this}, \pi) \wedge d(\pi, \tau) \rightarrow d(\text{this}, \tau)$
  $ant(\text{that}, \pi) \wedge d(\pi, \tau) \rightarrow d(\text{that}, \tau)$
- 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(that, (if((a woman)walk))((no man)talk))
- A refutation tree for *d*(true that, τ):

$$\leftarrow d(\text{true that}, \tau)$$
$$\downarrow$$
$$\leftarrow d(\text{that}, \tau)$$
$$\downarrow$$
$$\leftarrow ant(\text{that}, \pi),\ d(\pi, \tau)$$
$$\downarrow \ \pi := (\text{if((a woman)walk))((no man)talk)}$$
$$\leftarrow d((\text{if((a woman)walk))((no man)talk)}, \tau)$$
$$\vdots$$
$$\text{(continued as before)}$$

Liar and Truth-teller

Add *ant*(this, false this) to the database in case of the Liar; *ant*(this, true this) in case of the Truth-teller.

a.– Liar

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

b.– Truth-teller

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

Connections with Lambda Grammars/ACGs 1

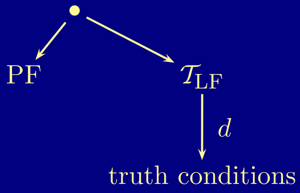
- There are two vocabularies, one consisting of all sans serif non-italic constants and one consisting of all constants.
- $\mathcal{T}^p_{\mathsf{LF}}$ is obtained by taking linear terms over the first vocabulary. Restriction to abstraction over type e variables here. Let $\mathcal{T}_{\mathsf{LF}}$ be the set of terms that is obtained by dropping the restriction to type p in the definition of $\mathcal{T}^p_{\mathsf{LF}}$.
- The family of functions $\{d^k\}_k$ associates elements of $\mathcal{T}_{\mathsf{LF}}$ with terms over the second vocabulary. Images may be non-linear (as in Lambda Grammars). The d^k may crucially be partial.

Connections with Lambda Grammars/ACGs 2

- Many clauses in the definition of the d^k are compatible with extending $\{d^k\}_k$ to a term homomorphism with underlying type homomorphism given by $e \mapsto e$, $p \mapsto st$.
- But, crucially, not all of them. Truly intentional predicates express a relation to a sense, not to the associated set of possible worlds. Jump to Meaning Postulates

A Possible Architecture of the Grammar

- A possible architecture:



- Each arrow a 'liberalized' term homomorphism.
- Do this with Lambda Grammars as collections of ACGs. (1-dimensional Lambda Grammars in original formulation cannot distinguish between abstract and concrete levels.)

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. In particular, 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.
- The match with Lambda Grammars and ACGs is not perfect but may become so if definitions can be liberalized without breaking too much.

Further Reading

Reinhard Muskens.
Sense and the Computation of Reference.
Linguistics and Philosophy, to appear.
http://let.uvt.nl/general/people/rmuskens