English as a programming language

Yiannis N. Moschovakis
UCLA and University of Athens

Tarski Lecture 2, March 5, 2008

Frege on sense

"*[the sense of a sign] may be the common property of many people*" Meanings are public (abstract?) objects

"*The sense of a proper name is grasped by everyone who is sufficiently familiar with the language . . . Comprehensive knowledge of the thing denoted . . . we never attain*"

Speakers of the language know the meanings of terms

"*The same sense has different expressions in different languages or even in the same language*"

"*The difference between a translation and the original text should properly not overstep the [level of the idea]*"

Faithful translation should preserve meaning

Outline of Lecture 2

Slogan:

The meaning of a term is the algorithm which computes its denotation

(1) Formal Fregean semantics in $\mathsf{L}_r^\lambda(K)$

(2) Meaning and synonymy in $\mathsf{L}_r^\lambda(K)$

(3) What are the objects of belief? (Local synonymy)

(4) The decision problem for synonymy

Sense and denotation as algorithm and value (1994)
A logical calculus of meaning and synonymy (2006)
Two aspects of situated meaning (with E. Kalyvianaki, to appear)
Posted in www.math.ucla.edu/∼ynm

The methodology of formal Fregean semantics

- An *interpreted formal language* L is selected
- The rendering operation on a fragment of English:

$$\text{English expression} + \text{informal context} \xrightarrow{\text{render}} \text{formal expression} + \text{state}$$

- Semantic values (denotations, meanings, etc.) are defined rigorously for the formal expressions of L and assigned to English expressions via the rendering operation

- Montague: L *should be a higher type language*
  (to interpret co-ordination, co-indexing, ...)
- Claim: L *should be a programming language*
  (to interpret self-reference and to define meanings properly)

The typed λ-calculus with recursion $\mathsf{L}_r^\lambda(K)$ - types

An extension of the typed λ-calculus, into which Montague's Language of Intensional Logic LIL can be easily interpreted (Gallin)

Basic types $b \equiv e \mid t \mid s$ (entities, truth values, states)

Types: $\sigma :\equiv b \mid (\sigma_1 \to \sigma_2)$

Abbreviation: $\sigma_1 \times \sigma_2 \to \tau \equiv (\sigma_1 \to (\sigma_2 \to \tau))$

Every non-basic type is uniquely of the form

$$\sigma \equiv \sigma_1 \times \cdots \times \sigma_n \to b$$

$\mathrm{level}(b) = 0$
$\mathrm{level}(\sigma_1 \times \cdots \times \sigma_n \to b) = \max\{\mathrm{level}(\sigma_1), \ldots, \mathrm{level}(\sigma_n)\} + 1$

The typed λ-calculus with recursion $\mathsf{L}^{\lambda}_{r}(K)$ - syntax

Pure variables: $v_0^\sigma, v_1^\sigma, \ldots$, for each type σ $(v : \sigma)$
Pure parameters: $\bar{u}$ for each state u (for convenience only)
Recursive variables: $p_0^\sigma, p_1^\sigma, \ldots$, for each type σ $(p : \sigma)$

Constants: A finite set K of typed constants (run, cow, he, the, every)

Terms – with assumed type restrictions and assigned types $(A : \sigma)$

$$A :\equiv v \mid \bar{u} \mid p \mid c \mid B(C) \mid \lambda(v)(B) \\ \mid A_0 \text{ where } \{p_1 = A_1, \ldots, p_n = A_n\}$$

$$C : \sigma, B : (\sigma \to \tau) \implies B(C) : \tau$$
$$v : \sigma, B : \tau \implies \lambda(v)(B) : (\sigma \to \tau)$$
$$A_0 : \sigma \implies A_0 \text{ where } \{p_1 = A_1, \ldots, p_n = A_n\} : \sigma$$

Abbreviation: $A(B, C, D) \equiv A(B)(C)(D)$

$\mathsf{L}^\lambda_r(K)$ - denotational semantics

- We are given basic sets $\mathbb{T}_s, \mathbb{T}_e$ and $\mathbb{T}_t \subseteq \mathbb{T}_e$ for the basic types

$$\mathbb{T}_{\sigma\to\tau} = \text{the set of all functions } f : \mathbb{T}_\sigma \to \mathbb{T}_\tau$$
$$\mathbb{P}_b = \mathbb{T}_b \cup \{\bot\} = \text{the ``flat poset'' of } \mathbb{T}_b$$
$$\mathbb{P}_{\sigma\to\tau} = \text{the set of all functions } f : \mathbb{T}_\sigma \to \mathbb{P}_\tau$$

$\mathbb{T}_\sigma \subseteq \mathbb{P}_\sigma$ and $\mathbb{P}_\sigma$ is a complete poset (with the pointwise ordering)

- We are given an object $c : \mathbb{P}_\sigma$ for each constant $\mathsf{c} : \sigma$

- Pure variables of type σ vary over $\mathbb{T}_\sigma$; recursive ones over $\mathbb{P}_\sigma$
- If $A : \sigma$ and π is a type-respecting assignment to the variables, then $\mathrm{den}(A)(\pi) \in \mathbb{P}_\sigma$
- Recursive terms are interpreted by the taking of least-fixed-points

Rendering natural language in $\mathsf{L}^{\lambda}_{r}(K)$

$$\tilde{t} \equiv (s \to t) \qquad \text{(type of Carnap intensions)}$$

$$\tilde{e} \equiv (s \to e) \qquad \text{(type of individual concepts)}$$

$$\text{Abelard loves Eloise} \xrightarrow{\text{render}} \text{loves(Abelard,Eloise)} : \tilde{t}$$

$$\text{Bush is the president} \xrightarrow{\text{render}} \text{eq(Bush,the(president))} : \tilde{t}$$

$$\text{liar} \xrightarrow{\text{render}} p \text{ where } \{p = \neg p\} : t$$

$$\text{truthteller} \xrightarrow{\text{render}} p \text{ where } \{p = p\} : t$$

$$\text{Abelard}, \text{Eloise}, \text{Bush} : \tilde{e}$$

$$\text{president} : \tilde{e} \to \tilde{t}, \text{eq} : \tilde{e} \times \tilde{e} \to \tilde{t}$$

$$\neg : t \to t, \text{the} : (\tilde{e} \to \tilde{t}) \to \tilde{e}$$

$$\text{den(liar)} = \text{den(truthteller)} = \bot$$

Co-ordination and co-indexing in $\mathsf{L}_r^\lambda(K)$

John stumbled and fell vs. John stumbled and he fell

John stumbled and fell $\xrightarrow{\text{render}} \lambda(x)\Big(\text{stumbled}(x)\ \&\ \text{fell}(x)\Big)(\text{John})$

(predication after co-ordination)

This is in Montague's LIL (as it is interpreted in $\mathsf{L}_r^\lambda(K)$)

John stumbled and he fell $\xrightarrow{\text{render}} \text{stumbled}(j)\ \&\ \text{fell}(j) \text{ where } \{j = \text{John}\}$

(conjunction after co-indexing)

The logical form of this sentence <u>cannot</u> be captured faithfully in LIL — recursion models co-indexing preserving logical form

Can we say nonsense in $\mathsf{L}_r^\lambda(K)$?

Yes!
In particular, we have parameters over states—so we can explicitly refer to the state (even to two states in one term); LIL does not allow this, *because we cannot do this in English*

Consider the terms

$$A \equiv \mathsf{rapidly}(\mathsf{tall})(\mathsf{John}),\ \ B \equiv \mathsf{rapidly}(\mathsf{sleeping})(\mathsf{John}) : \tilde{t}$$

A and B are terms of LIL,

not the renderings of correct English sentences

- The target formal language is a tool for defining rigorously the desired semantic values and it needs to be richer than a direct formalization of the relevant fragment of English

—to insure compositionality, if for no other reason

Meaning and synonymy in $\mathsf{L}^{\lambda}_{r}(K)$

- For a sentence $A : \tilde{t}$, the Montague sense of A is $\mathrm{den}(A) : \mathbb{T}_s \to \mathbb{T}_t$, so that

  there are infinitely many primes

  is Montague-synonymous with $1 + 1 = 2$

- In $\mathsf{L}^{\lambda}_{r}(K)$: *The meaning of a term A is modeled by an algorithm* $\mathrm{int}(A)$ *which computes den*$(A)(\pi)$ *for every* π
- The referential intension $\mathrm{int}(A)$ is compositionally determined from A
- $\mathrm{int}(A)$ is an abstract (not necessarily implementable) recursive algorithm of $\mathsf{L}^{\lambda}_{r}(K)$
- Referential synonymy: $\boxed{A \approx B \iff \mathrm{int}(A) \sim \mathrm{int}(A)}$

Reduction, Canonical Forms and the Synonymy Theorem

- A reduction relation $A \Rightarrow B$ is defined on terms of $\mathsf{L}_r^\lambda(K)$
- Each term A is reducible to a unique (up to congruence) irreducible recursive term, its canonical form

$$A \Rightarrow \mathrm{cf}(A) \equiv A_0 \text{ where } \{p_1 = A_1, \ldots, p_n = A_n\}$$

- $\boxed{\mathrm{int}(A) = (\mathrm{den}(A_0), \mathrm{den}(A_1), \ldots, \mathrm{den}(A_n))}$
- The parts $A_0, \ldots, A_n$ of A are irreducible, explicit terms
- $\mathrm{cf}(A)$ models the logical form of A
- *Synonymy Theorem. $A \approx B$ if and only if*

$$B \Rightarrow \mathrm{cf}(B) \equiv B_0 \text{ where } \{p_1 = B_1, \ldots, p_m = B_m\}$$

so that $n = m$ and for $i \leq n$, $den(A_i) = den(B_i)$

Is this notion of meaning Fregean?

Evans (in a discussion of Dummett's similar, computational interpretations of Frege's sense):

> *"This leads [Dummett] to think generally that the sense of an expression is (not a way of thinking about its [denotation], but) a method or procedure for determining its denotation. So someone who grasps the sense of a sentence will be possessed of some method for determining the sentence's truth value*
> *. . . ideal verificationism*
> *. . . there is scant evidence for attributing it to Frege"*

Converse question: For a sentence A, if you possess the method determined by A for determining its truth value, do you then "grasp" the sense of A?

(Sounds more like Davidson rather than Frege)

The reduction calculus

$$\text{Bush is the president} \xrightarrow{\text{render}} \mathrm{eq}(\mathrm{Bush})(\mathrm{the}(\mathrm{president}))$$

$$\Rightarrow \mathrm{eq}(\mathrm{Bush})(L) \text{ where } \{L = \mathrm{the}(\mathrm{president})\}$$

$$\Rightarrow \mathrm{eq}(\mathrm{Bush})(L) \text{ where } \{L = \mathrm{the}(p) \text{ where } \{p = \mathrm{president}\}\}$$

$$\Rightarrow \mathrm{eq}(\mathrm{Bush})(L) \text{ where } \{L = \mathrm{the}(p), p = \mathrm{president}\}$$

$$\Rightarrow \Big(\mathrm{eq}(b) \text{ where } \{b = \mathrm{Bush}\}\Big)(L) \text{ where } \{L = \mathrm{the}(p), \\ p = \mathrm{president}\}$$

$$\Rightarrow \Big(\mathrm{eq}(b)(L) \text{ where } \{b = \mathrm{Bush}\}\Big) \text{ where } \{L = \mathrm{the}(p), \\ p = \mathrm{president}\}$$

$$\Rightarrow_{cf} \boxed{\mathrm{eq}(b)(L) \text{ where } \{b = \mathrm{Bush}, L = \mathrm{the}(p), p = \mathrm{president}\}}$$

$$\text{He is the president} \xrightarrow{\text{render}} \mathrm{eq}(\mathrm{He})(\mathrm{the}(\mathrm{president}))$$

$$\Rightarrow_{cf} \boxed{\mathrm{eq}(b)(L) \text{ where } \{b = \mathrm{He}, L = \mathrm{the}(p), p = \mathrm{president}\}}$$

The reduction calculus

John loves and honors his father

$$\xrightarrow{\text{render}} \Big(\lambda(x)(\text{loves}(j,x)\ \&\ \text{honors}(j,x))\Big)(\text{father}(j)) \text{ where } \{j = \text{John}\}$$

$$\Rightarrow \Big[\Big(\lambda(x)(\text{loves}(j,x)\ \&\ \text{honors}(j,x))\Big)(f) \text{ where } \{f = \text{father}(j)\}\Big]$$
$$\text{where } \{j = \text{John}\}$$

$$\Rightarrow \Big(\lambda(x)(\text{loves}(j,x)\ \&\ \text{honors}(j,x))\Big)(f)$$
$$\text{where } \{f = \text{father}(j), j = \text{John}\}$$

$$\Rightarrow \Big(\lambda(x)\Big[(l\ \&\ h) \text{ where } \{l = \text{loves}(j,x), h = \text{honors}(j,x)\}\Big]\Big)(f)$$
$$\text{where } \{f = \text{father}(j), j = \text{John}\}$$

$$\Rightarrow \Big(\lambda(x)(l(x)\ \&\ h(x))$$
$$\text{where } \{l = \lambda(x)\text{loves}(j,x), h = \lambda(x)\text{honors}(j,x)\}\Big)(f)$$
$$\text{where } \{f = \text{father}(j), j = \text{John}\}$$

$$\Rightarrow \lambda(x)(l(x)\ \&\ h(x))(f)$$
$$\text{where } \{l = \text{loves}(j,\cdot), h = \text{honors}(j,\cdot), f = \text{father}(j), j = \text{John}\}$$

Utterances, local meanings, local synonymy

An utterance is a pair (A, u), where A is a sentence, $A : \tilde{t}$ and u is a state; it is expressed in $\mathrm{L}^{\lambda}_{r}(K)$ by the term $A(\bar{u})$

The local meaning of A at the state u is $\mathrm{int}(A(\bar{u}))$

$$A \approx_u B \iff A(\bar{u}) \approx B(\bar{u})$$

Bush is the president$(\bar{u})$

$\Rightarrow_{cf} \mathrm{eq}(b)(L)(\bar{u})$ where $\{b = \mathrm{Bush}, L = \mathrm{the}(p), p = \mathrm{president}\}$

He is the president$(\bar{u})$

$\Rightarrow_{cf} \mathrm{eq}(b)(L)(\bar{u})$ where $\{b = \mathrm{He}, L = \mathrm{the}(p), p = \mathrm{president}\}$

Bush is the president $\not\approx_u$ He is the president

even if at the state $\bar{u}$, $\mathrm{He}(\bar{u}) = \mathrm{Bush}(\bar{u})$

Three aspects of meaning for a sentence $A : \tilde{t}$

Referential intension	$\mathrm{int}(A)$	Referential synonymy $\approx$
Local meaning at u	$\mathrm{int}(A(\bar{u}))$	Local synonymy $\approx_u$
Factual content at u	$\mathrm{FC}(A, u)$	Factual synonymy $\approx_{f,u}$

The *factual content* of a sentence at a state u gives a *representation of the world* at u (Eleni Kalyvianaki's Ph.D. Thesis)

Bush is the president $\not\approx_u$ He is the president

Bush is the president $\approx_{f,u}$ He is the president

Claim: *The objects of belief are local meanings*

The distinction between local meaning and factual content are related to David Kaplan's distinction between the *character* and *content* of a sentence at a state

Some referential (global) synonymies and non-synonymies

- There are infinitely many primes $\not\approx 1 + 1 = 2$
- $A \mathbin{\&} B \approx B \mathbin{\&} A$
- The morning star is the evening star

  $\approx$The evening star is the morning star

  (This fails with Montague's renderings)
- Abelard loves Eloise $\approx$ Eloise $\boxed{\text{is loved by}}$ Abelard (Frege)
- $2 + 3 = 6 \approx 3 + 2 = 6$ (with $+$ and the numbers primitive)
- liar $\not\approx$ truthteller
- John stumbled and he fell $\xrightarrow{\text{render}}$

  $A \equiv \text{stumbled}(j) \mathbin{\&} \text{fell}(j) \text{ where } \{j = \text{John}\}$

  A is not $\approx$ with any *explicit* term (including any term from LIL)

Is referential synonymy decidable?

Synonymy Theorem. $A \approx B$ if and only if

$$A \Rightarrow \mathrm{cf}(A) \equiv A_0 \text{ where } \{p_1 = A_1, \dots, p_n = A_n\}$$
$$B \Rightarrow \mathrm{cf}(B) \equiv B_0 \text{ where } \{p_1 = B_1, \dots, p_n = B_n\}$$

so that for $i = 0, \dots, n$ and all π, $den(A_i)(\pi) = den(B_i)(\pi)$.

- Synonymy is reduced to denotational equality for explicit, irreducible terms (the truth facts of A)
- Denotational equality for arbitrary terms is undecidable (there are constants, with fixed interpretations)
- The explicit, irreducible terms are very special

— but by no means trivial!

The synonymy problem for $\mathsf{L}^\lambda_r(K)$ (with finite K)

- The decision problem for $\mathsf{L}^\lambda_r(K)$-synonymy is open

Theorem *If the set of constants K is finite, then synonymy is decidable for terms of* **adjusted level** ≤ 2

These include terms constructed "simply" from

Names of "pure" objects	$0, 1, 2, \emptyset, \ldots$	$: e$
Names, demonstratives	$\mathrm{John}, \mathrm{I}, \mathrm{he}, \mathrm{him}$	$: \tilde{e}$
Common nouns	$\mathrm{man}, \mathrm{unicorn}, \mathrm{temperature}$	$: \tilde{e} \to \tilde{t}$
Adjectives	$\mathrm{tall}, \mathrm{young}$	$: (\tilde{e} \to \tilde{t}) \to (\tilde{e} \to \tilde{t})$
Propositions	it rains	$: \tilde{t}$
Intransitive verbs	$\mathrm{stand}, \mathrm{run}, \mathrm{rise}$	$: \tilde{e} \to \tilde{t}$
Transitive verbs	$\mathrm{find}, \mathrm{loves}, \mathrm{be}$	$: \tilde{e} \times \tilde{e} \to \tilde{t}$
Adverbs	$\mathrm{rapidly}$	$: (\tilde{e} \to \tilde{t}) \to (\tilde{e} \to \tilde{t})$

Proof is by reducing this claim to the Main Theorem in the 1994 paper (for a corrected version see www.math.ucla.edu/$\sim$ynm)

Explicit, irreducible identities that must be known

- Los Angeles = LA   (Athens = Αθήνα)
- $x \mathbin{\&} y = y \mathbin{\&} x$
- $\text{between}(x, y, z) = \text{between}(x, z, y)$
- $\text{love}(x, y) = \text{be\_loved}(y, x)$

A dictionary is needed—but what kind and how large?

$$\text{ev}_2(\lambda(u_1, u_2)r(u_1, u_2, \vec{a}), b, z) = \text{ev}_1(\lambda(v)r(v, z, \vec{a}), b)$$

Evaluation functions: both sides are equal to $r(b, z, \vec{a})$

The dictionary line which determines this is (essentially)

$$\lambda(s)x(s, z) = \lambda(s)y(s) \implies \text{ev}_2(x, b, z) = \text{ev}_1(y, b)$$

The form of the decision algorithm

- A finite list of true dictionary lines is constructed, which codifies the relationships between the constants
- Given two explicit, irreducible terms A, B of adjusted level ≤ 2, we construct (effectively) a finite set $L(A, B)$ of lines such that

$$\models A = B$$
$$\iff \text{every line in } L(A,B) \text{ is congruent to one in the dictionary}$$

- It is a lookup algorithm, justified by a finite basis theorem
- Complexity: NP; the graph isomorphism problem is reducible to the synonymy problem for very simple (propositional) recursive terms