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 $L^\lambda_r(K)$
(2) Meaning and synonymy in $L^\lambda_r(K)$
(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: $L$ \textit{should be a higher type language}
  (to interpret co-ordination, co-indexing, \ldots)

- Claim: $L$ \textit{should be a programming language}
  (to interpret self-reference and to define meanings properly)
The typed $\lambda$-calculus with recursion $L^\lambda_r(K)$ - types

An extension of the typed $\lambda$-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:* $\sigma \equiv b \mid (\sigma_1 \rightarrow \sigma_2)$

*Abbreviation:* $\sigma_1 \times \sigma_2 \rightarrow \tau \equiv (\sigma_1 \rightarrow (\sigma_2 \rightarrow \tau))$

Every non-basic type is uniquely of the form

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

level($b$) = 0
level($\sigma_1 \times \cdots \times \sigma_n \rightarrow b$) = max{level($\sigma_1$), \ldots, level($\sigma_n$)} + 1
The typed $\lambda$-calculus with recursion $L^\lambda_r(K)$ - syntax

**Pure variables:** $v_0^\sigma, v_1^\sigma, \ldots$, for each type $\sigma$ ($v : \sigma$)

**Pure parameters:** $\bar{u}$ for each state $u$ (for convenience only)

**Recursive variables:** $p_0^\sigma, p_1^\sigma, \ldots$, for each type $\sigma$ ($p : \sigma$)

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

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

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

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

Abbreviation: $A(B, C, D) \equiv A(B)(C)(D)$
\[ L^\lambda_r(K) \] - denotational semantics

- We are given basic sets \( T_s, T_e \) and \( T_t \subseteq T_e \) for the basic types

\[ T_{\sigma \rightarrow \tau} = \text{the set of all functions } f : T_\sigma \rightarrow T_\tau \]
\[ P_b = T_b \cup \{ \bot \} = \text{the “flat poset” of } T_b \]
\[ P_{\sigma \rightarrow \tau} = \text{the set of all functions } f : T_\sigma \rightarrow P_\tau \]

\( T_\sigma \subseteq P_\sigma \) and \( P_\sigma \) is a complete poset (with the pointwise ordering)

- We are given an object \( c : P_\sigma \) for each constant \( c : \sigma \)

- Pure variables of type \( \sigma \) vary over \( T_\sigma \); recursive ones over \( P_\sigma \)
- If \( A : \sigma \) and \( \pi \) is a type-respecting assignment to the variables, then \( \text{den}(A)(\pi) \in P_\sigma \)
- Recursive terms are interpreted by the taking of least-fixed-points
Rendering natural language in $L^\lambda_r(K)$

$\tilde{t} \equiv (s \rightarrow t)$ \hspace{1cm} (type of Carnap intensions)

$\tilde{e} \equiv (s \rightarrow e)$ \hspace{1cm} (type of individual concepts)

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

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

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

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

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

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

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

$\text{den(liar)} = \text{den(truth teller)} = \bot$
Co-ordination and co-indexing in $L_\lambda^r(K)$

John stumbled and fell vs. John stumbled and he fell

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

(predication after co-ordination)

This is in Montague’s LIL (as it is interpreted in $L_\lambda^r(K)$)

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

(conjunction after co-indexing)

The logical form of this sentence cannot be captured faithfully in LIL — recursion models co-indexing preserving logical form
Can we say nonsense in $L^\lambda_r(K)$?

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

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

$A$ 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 $L^\lambda_r(K)$

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

  there are infinitely many primes

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

- In $L^\lambda_r(K)$: The meaning of a term $A$ is modeled by an algorithm $\text{int}(A)$ which computes $\text{den}(A)(\pi)$ for every $\pi$

- The referential intension $\text{int}(A)$ is compositionally determined from $A$

- $\text{int}(A)$ is an abstract (not necessarily implementable) recursive algorithm of $L^\lambda_r(K)$

- Referential synonymy: $A \approx B \iff \text{int}(A) \sim \text{int}(A)$
Reduction, Canonical Forms and the Synonymy Theorem

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

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

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

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

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

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

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

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

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

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

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

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

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

$\Rightarrow_{\text{cf}} \text{eq}(b)(L) \text{ where } \{b = \text{He}, L = \text{the}(p), p = \text{president}\}$
The reduction calculus

John loves and honors his father

\[ \text{render} \quad \left( \lambda(x)(\text{loves}(j, x) \ \& \ \text{honors}(j, x)) \right)(\text{father}(j)) \text{ where } \{ j = \text{John} \} \]

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

\[ \Rightarrow \left( \lambda(x)(\text{loves}(j, x) \ \& \ \text{honors}(j, x)) \right)(f) \]

\[ \Rightarrow \left( \lambda(x)[(l \ \& \ h) \text{ where } \{ l = \text{loves}(j, x), h = \text{honors}(j, x) \}] \right)(f) \]

\[ \Rightarrow \left( \lambda(x)(l(x) \ \& \ h(x)) \right) \]

\[ \Rightarrow \lambda(x)(l(x) \ \& \ h(x))(f) \]

where \[ \{ l = \lambda(x)\text{loves}(j, x), h = \lambda(x)\text{honors}(j, x) \} \]

where \[ \{ f = \text{father}(j), j = \text{John} \} \]
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 \(L_r^\lambda(K)\) by the term \(A(\tilde{u})\)

The local meaning of \(A\) at the state \(u\) is \(\text{int}(A(\tilde{u}))\)

\[
A \approx_u B \iff A(\tilde{u}) \approx B(\tilde{u})
\]

Bush is the president(\(\tilde{u}\))
\[
\Rightarrow_{cf} \text{eq}(b)(L)(\tilde{u}) \quad \text{where} \quad \{b = \text{Bush}, L = \text{the}(p), p = \text{president}\}
\]

He is the president(\(\tilde{u}\))
\[
\Rightarrow_{cf} \text{eq}(b)(L)(\tilde{u}) \quad \text{where} \quad \{b = \text{He}, L = \text{the}(p), p = \text{president}\}
\]

Bush is the president \(\not\approx_u\) He is the president

even if at the state \(\tilde{u}\), \(\text{He}(\tilde{u}) = \text{Bush}(\tilde{u})\)
Three aspects of meaning for a sentence $A : \tilde{t}$

- Referential intension $\text{int}(A)$
- Referential synonymy $\approx$
- Local meaning at $u$ $\text{int}(A(\bar{u}))$
- Local synonymy $\approx_u$
- Factual content at $u$ $\text{FC}(A, u)$
- 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 $\not\approx 1 + 1 = 2$
- $A \& B \approx B \& A$
- 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 is loved by Abelard (Frege)
- $2 + 3 = 6 \approx 3 + 2 = 6$ (with + and the numbers primitive)
- liar $\not\approx$ truth teller
- John stumbled and he fell
  \[
  A \equiv \text{stumbled}(j) \& \text{fell}(j) \text{ where } \{j = \text{John}\}
  \]
  $A$ 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

\[
A \Rightarrow \text{cf}(A) \equiv A_0 \text{ where } \{p_1 = A_1, \ldots, p_n = A_n\} \\
B \Rightarrow \text{cf}(B) \equiv B_0 \text{ where } \{p_1 = B_1, \ldots, p_n = B_n\}
\]

so that for \( i = 0, \ldots, n \) and all \( \pi \), \( \text{den}(A_i)(\pi) = \text{den}(B_i)(\pi) \).

- 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 $L_r^\lambda(K)$ (with finite $K$)

- The decision problem for $L_r^\lambda(K)$-synonymy is open

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

These include terms constructed “simply” from

Names of “pure” objects $0, 1, 2, \emptyset, \ldots : e$
Names, demonstratives John, I, he, him : ē
Common nouns man, unicorn, temperature : ē $\rightarrow$ ù
Adjectives tall, young : $(ē \rightarrow ū) \rightarrow (ē \rightarrow ū)$
Propositions it rains : ū
Intransitive verbs stand, run, rise : ē $\rightarrow$ ū
Transitive verbs find, loves, be : ē $\times$ ē $\rightarrow$ ū
Adverbs rapidly : $(ē \rightarrow ū) \rightarrow (ē \rightarrow ū)$

Proof is by reducing this claim to the Main Theorem in the 1994 paper (for a corrected version see www.math.ucla.edu/~ynm)
Explicit, irreducible identities that must be known

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

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

\[
ev_2(\lambda(u_1, u_2)r(u_1, u_2, \vec{a}), b, z) = \ev_1(\lambda(v)r(v, z, \vec{a}), b)
\]

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

The dictionary line which determines this is (essentially)

\[
\lambda(s)x(s, z) = \lambda(s)y(s) \implies \ev_2(x, b, z) = \ev_1(y, b)
\]
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 $\leq 2$, we construct (effectively) a finite set $L(A, B)$ of lines such that

$$|\models A = B$$

$$\iff$$ every line in $L(A, B)$ 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