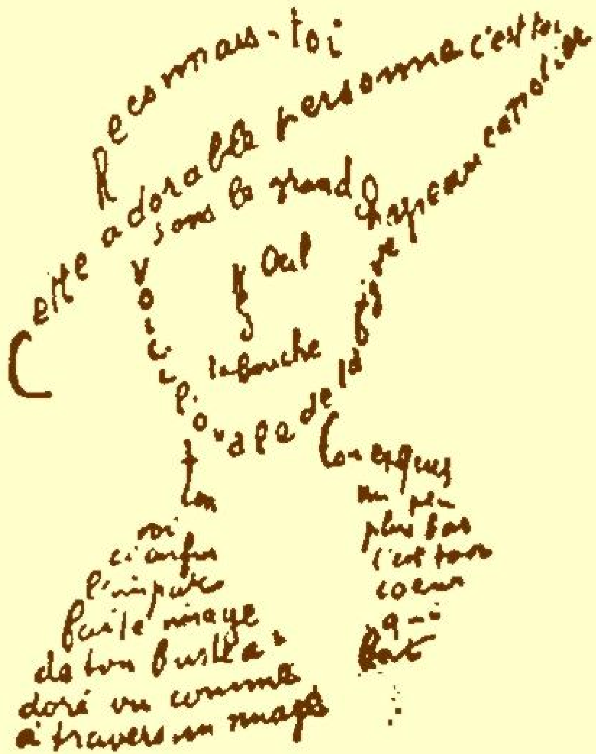
Reconnais-toi
Cette adorable personne c'est toi
sous le grand chapeau canotier
Oeil
la bouche
Voici l'ovale de ta figure
ton cou exquis
un peu plus bas c'est ton coeur qui bat
voici enfin l'imparfaite image de ton buste adoré vu comme à travers un nuage

Logical Semantics

I. Montague Semantics

There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory. On this point I differ from a number of philosophers (...).

R. Montague,
Universal Grammar,
Theoria 36:373–398 (1970)

Montague's legacy

- The notion of fragment.
- Semantics as an homomorphic image of syntax.
- Semantic interpretation through a translation into an intermediate logical form.

A direct naive interpretation

S $\to$ NP VP	$[\![\text{S}]\!] = [\![\text{VP}]\!]\,[\![\text{NP}]\!]$
VP $\to$ tV NP	$[\![\text{VP}]\!] = [\![\text{tV}]\!]\,[\![\text{NP}]\!]$
tV $\to$ loves	$[\![\text{tV}]\!] = \lambda y.\,\lambda x.\,\mathbf{love}\,y\,x$
NP $\to$ John	$[\![\text{NP}]\!] = \mathbf{j}$
NP $\to$ Mary	$[\![\text{NP}]\!] = \mathbf{m}$

where:

$\mathbf{j}, \mathbf{m} : \iota$

$\mathbf{love} : \iota \to \iota \to o$

Quantified noun phrases

S → NP VP	$[\![\text{S}]\!] = [\![\text{VP}]\!]\,[\![\text{NP}]\!]$
VP → tV NP	$[\![\text{VP}]\!] = [\![\text{tV}]\!]\,[\![\text{NP}]\!]$
tV → loves	$[\![\text{tV}]\!] = \lambda o.\,\lambda s.\,s\,(\lambda x.\,o\,(\lambda y.\,\mathbf{love}\,x\,y))$
NP → John	$[\![\text{NP}]\!] = \lambda k.\,k\,\mathbf{j}$
NP → somebody	$[\![\text{NP}]\!] = \lambda k.\,\exists x.\,k\,x$

Nouns, adjectives, and determiners

S → NP VP	$[\![\text{S}]\!] = [\![\text{VP}]\!]\,[\![\text{NP}]\!]$
VP → tV NP	$[\![\text{VP}]\!] = [\![\text{tV}]\!]\,[\![\text{NP}]\!]$
NP → Det N	$[\![\text{NP}]\!] = [\![\text{Det}]\!]\,[\![\text{N}]\!]$
N → Adj N	$[\![\text{N}]\!] = [\![\text{Adj}]\!]\,[\![\text{N}]\!]$
tV → loves	$[\![\text{tV}]\!] = \lambda o.\,\lambda s.\,s\,(\lambda x.\,o\,(\lambda y.\,\mathbf{love}\,x\,y))$
NP → John	$[\![\text{NP}]\!] = \lambda k.\,k\,\mathbf{j}$
NP → somebody	$[\![\text{NP}]\!] = \lambda k.\,\exists x.\,k\,x$
N → woman	$[\![\text{N}]\!] = \lambda x.\,\mathbf{woman}\,x$
N → man	$[\![\text{N}]\!] = \lambda x.\,\mathbf{man}\,x$
Adj → nice	$[\![\text{Adj}]\!] = \lambda n.\,\lambda x.\,n\,x \wedge \mathbf{nice}\,x$
Det → every	$[\![\text{Det}]\!] = \lambda n.\,\lambda m.\,\forall x.\,n\,x \supset m\,x$
Det → a	$[\![\text{Det}]\!] = \lambda n.\,\lambda m.\,\exists x.\,n\,x \wedge m\,x$

where:

$$\mathbf{woman}, \mathbf{man}, \mathbf{nice} : \iota \rightarrow o$$

Relative clauses

N $\rightarrow$ N rC	$[\![\text{N}]\!] = [\![\text{RC}]\!]\,[\![\text{N}]\!]$
rC $\rightarrow$ rP VP	$[\![\text{RC}]\!] = [\![\text{RP}]\!]\,[\![\text{VP}]\!]$
rP $\rightarrow$ who	$[\![\text{rP}]\!] = \lambda r.\,\lambda n.\,\lambda x.\,n\,x \wedge r\,(\lambda k.\,k\,x)$

The categorial syntax/semantics interface

loves	:	$(NP \setminus S)/NP$
John	:	NP
Mary	:	NP
somebody	:	NP
woman	:	N
man	:	N
nice	:	N/N
every	:	NP/N
a	:	NP/N
who	:	$(N \setminus N)/(NP \setminus S)$

$$
\begin{aligned}
[\![S]\!] &= o \\
[\![NP]\!] &= (\iota \rightarrow o) \rightarrow o \\
[\![N]\!] &= \iota \rightarrow o \\
[\![\alpha \setminus \beta]\!] &= [\![\alpha]\!] \rightarrow [\![\beta]\!] \\
[\![\beta / \alpha]\!] &= [\![\alpha]\!] \rightarrow [\![\beta]\!]
\end{aligned}
$$

Scope ambiguities

Every man loves a woman

$$\forall x.\mathbf{man}\,x \supset (\exists y.\mathbf{woman}\,y \wedge \mathbf{love}\,x\,y)$$
$$\exists y.\mathbf{woman}\,y \wedge (\forall x.\mathbf{man}\,x \wedge \mathbf{love}\,x\,y)$$

Subject wide scope:

$$\lambda o.\,\lambda s.\,s\,(\lambda x.\,o\,(\lambda y.\,\mathbf{love}\,x\,y))$$

Object wide scope:

$$\lambda o.\,\lambda s.\,o\,(\lambda y.\,s\,(\lambda x.\,\mathbf{love}\,x\,y))$$

Another solution:

$$\begin{array}{rcl} \text{every} & : & (\mathsf{S}/(\mathsf{NP} \setminus \mathsf{S}))/\mathsf{N} \\ \text{a} & : & ((\mathsf{S}/\mathsf{NP}) \setminus \mathsf{S})/\mathsf{N} \end{array}$$

with

$$\begin{array}{rcl} [\![\mathsf{S}]\!] & = & o \\ [\![\mathsf{NP}]\!] & = & \iota \end{array}$$

Some intensional puzzles

De re and *de dicto*

John seeks a unicorn

Intersective and non-intersective adjectives

John is a french cook

John is an alleged cook

Partee's puzzle

The temperature is ninety. The temperature rises.

Montague's intentional logic

Higher-Order Classical Logic + modalities (necessity, past, and future).

One additional base type s, with the following associated terms, typing rules, and reduction rules:

$$\frac{\Gamma \vdash t : \alpha}{\Gamma \vdash {}^{\wedge}t : s \to \alpha} \qquad \frac{\Gamma \vdash t : s \to \alpha}{\Gamma \vdash {}^{\vee}t : \alpha}$$

$${}^{\vee}{}^{\wedge}t \to t$$

More about time and tense

Reichenbach's three points of time: point of speech (S), point of event (E), and point of reference (R).

tense	example	structure
pluperfect	I had seen	E-R-S
simple past	I saw	E,R-S
future in the past	I would see	R-E-S / R-E,S / R-S-E
present perfect	I have seen	E-S,R
present	I see	E,S,R

Remember hybrid logic:

$$\mathsf{P}(\downarrow r.\,@_r\mathsf{P}\phi) \quad \mathsf{P}(\downarrow r.\,@_r\phi) \quad \mathsf{P}(\downarrow r.\,@_r\mathsf{F}\phi) \quad \mathsf{P}\phi \quad \phi$$

II. Continuations

Invented to provide programming languages control operators (exit, goto,...) with a compositional semantics.

Continuation passing style (call by value)

1. $\overline{c} \;=\; \lambda k.\, k\, c$
2. $\overline{x} \;=\; \lambda k.\, k\, x$
3. $\overline{\lambda x.\, M} \;=\; \lambda k.\, k\,(\lambda x.\, \overline{M})$
4. $\overline{M\, N} \;=\; \lambda k.\, \overline{M}\,(\lambda m.\, \overline{N}\,(\lambda n.\, m\, n\, k))$

$\overline{\alpha} = \neg\neg\alpha^*$, where:

1. $\bot^* \;=\; \bot$
2. $a^* \;=\; a,$ for a atomic
3. $(\alpha \to \beta)^* \;=\; \alpha^* \to \overline{\beta}$

III. Abstract Categorial Grammars

Types, signatures and λ-terms:

$\mathcal{T}(A)$ is the set of linear implicative types built on the set of atomic types A:

$$\mathcal{T}(A) \ ::= \ A \mid (\mathcal{T}(A) \multimap \mathcal{T}(A))$$

A higher-order linear signature is a triple $\Sigma = \langle A, C, \tau \rangle$, where:

- A is a finite set of atomic types;
- C is a finite set of constants;
- $\tau : C \rightarrow \mathcal{T}(A)$ is a function that assigns each constant in C with a linear implicative type built on A.

$\Lambda(\Sigma)$ denotes the set of linearλ-terms built upon a higher-order linear signature Σ.

Vocabularies and Lexicons:

A vocabulary is simply defined to be a higher-order linear signature.

Given two vocabularies $\mathbf{\Sigma}_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\mathbf{\Sigma}_2 = \langle A_2, C_2, \tau_2 \rangle$, a lexicon $\mathcal{L} = \langle F, G \rangle$ from $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_2$ is made of two functions:

$F : A_1 \rightarrow \mathcal{T}(A_2)$,

$G : C_1 \rightarrow \Lambda(\Sigma_2)$,

such that

$$\vdash_{\Sigma_2} G(c) : \widehat{F}(\tau_1(c)).$$

Definition:

An abstract categorial grammar is a quadruple

$$\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$$

where :

$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ are two higher-order linear signatures; Σ_1 is called the abstract vocabulary and Σ_2 is called the object vocabulary;

$\mathcal{L} : \Sigma_1 \rightarrow \Sigma_2$ is a lexicon from the abstract vocabulary to the object vocabulary;

$s \in \mathcal{T}(A_1)$ is a type of the abstract vocabulary; it is called the distinguished type of the grammar.

Languages generated by an ACG:

The abstract language generated by $\mathcal{G}$ ($\mathcal{A}(\mathcal{G})$) is defined as follows:

$$\mathcal{A}(\mathcal{G}) = \{t \in \Lambda(\Sigma_1) \mid \vdash_{\Sigma_1} t : s \text{ is derivable}\}$$

The object language generated by $\mathcal{G}$ ($\mathcal{O}(\mathcal{G})$) is defined to be the image of the abstract language by the term homomorphism induced by the lexicon $\mathcal{L}$:

$$\mathcal{O}(\mathcal{G}) = \{t \in \Lambda(\Sigma_2) \mid \exists u \in \mathcal{A}(\mathcal{G}).\ t = \mathcal{L}(u)\}$$

Strings as linear λ-terms

There is a canonical way of representing strings as linear λ-terms. It consists of representing strings as function composition:

$$`abbac' = \lambda x.\, a\,(b\,(b\,(a\,(c\,x))))$$

In this setting:

$$\begin{array}{rcl} \epsilon & \stackrel{\triangle}{=} & \lambda x.\, x \\ \alpha + \beta & \stackrel{\triangle}{=} & \lambda \alpha.\, \lambda \beta.\, \lambda x.\, \alpha\,(\beta\, x) \end{array}$$

Example

Pierre lit un article que Marie a écrit

$$\Sigma_0: \begin{array}{rcl} N, NP, S & : & \text{type;} \\ \mathsf{P}, \mathsf{M} & : & NP \\ \mathsf{A} & : & N \\ \mathsf{L}, \mathsf{AE} & : & NP \multimap (NP \multimap S) \\ \mathsf{U} & : & N \multimap NP \\ \mathsf{Q} & : & (NP \multimap S) \multimap (N \multimap N) \end{array}$$

Σ_1: /Pierre/, /Marie/, /article/, /lit/, /a écrit/, /un/, /que/ : $STRING$

Σ_2:

$$
\begin{array}{rcl}
\iota, o & : & \text{type}; \\
\mathbf{p}, \mathbf{m} & : & \iota \\
\text{article} & : & \iota \multimap o \\
\text{read}, \text{wrote} & : & \iota \multimap (\iota \multimap o) \\
\wedge & : & o \multimap (o \multimap o) \\
\exists & : & (\iota \rightarrow o) \multimap o
\end{array}
$$

$\mathcal{L}_1 : \Sigma_0 \rightarrow \Sigma_1$

$$N, NP, S \;:=\; STRING;$$

$$\begin{array}{rcll}
\mathsf{P} & := & /\mathsf{Pierre}/ & : \; NP \\
\mathsf{M} & := & /\mathsf{Marie}/ & : \; NP \\
\mathsf{A} & := & /\mathsf{article}/ & : \; N \\
\mathsf{L} & := & \lambda x.\,\lambda y.\,y + /\mathsf{lit}/ + x & : \; NP \multimap (NP \multimap S) \\
\mathsf{AE} & := & \lambda x.\,\lambda y.\,y + /\mathsf{a\ écrit}/ + x & : \; NP \multimap (NP \multimap S) \\
\mathsf{U} & := & \lambda x.\,/\mathsf{un}/ + x & : \; N \multimap NP \\
\mathsf{Q} & := & \lambda x.\,\lambda y.\,y + /\mathsf{que}/ + x\,\epsilon & : \; (NP \multimap S) \multimap (N \multimap N)
\end{array}$$

Parsing

/Pierre/ + /lit/ + /un/ + /article/ + /que/ + /Marie/ + /a écrit/

yields the following λ-term of type S:

$$\mathsf{L}\,(\mathsf{U}\,(\mathsf{Q}\,(\lambda x.\,\mathsf{AE}\,x\,\mathsf{M})\,\mathsf{A}))\,\mathsf{P}$$

$\mathcal{L}_2 : \Sigma_0 \rightarrow \Sigma_2$

$$
\begin{array}{rcl}
S & := & o; \\
N & := & \iota \multimap o; \\
NP & := & (\iota \multimap o) \multimap o;
\end{array}
$$

$$
\begin{array}{rcllcl}
\mathsf{P} & := & \lambda k.\, k\, \mathbf{p} & : & NP \\
\mathsf{M} & := & \lambda k.\, k\, \mathbf{m} & : & NP \\
\mathsf{A} & := & \lambda x.\, \mathsf{article}\, x & : & N \\
\mathsf{L} & := & \lambda p.\, \lambda q.\, p\, (\lambda x.\, q\, (\lambda y.\, \mathsf{read}\, y\, x)) & : & NP \multimap (NP \multimap S) \\
\mathsf{AE} & := & \lambda p.\, \lambda q.\, p\, (\lambda x.\, q\, (\lambda y.\, \mathsf{wrote}\, y\, x)) & : & NP \multimap (NP \multimap S) \\
\mathsf{U} & := & \lambda p.\, \lambda q.\, \exists x.\, (p\, x) \wedge (q\, x) & : & N \multimap NP \\
\mathsf{Q} & := & \lambda r.\, \lambda p.\, \lambda x.\, (p\, x) \wedge (r\, (\lambda k.\, k\, x)) & : & (NP \multimap P) \multimap (N \multimap N)
\end{array}
$$

Applying $\mathcal{L}_2$ to

$$\mathsf{L}\, (\mathsf{U}\, (\mathsf{Q}\, (\lambda x.\, \mathsf{AE}\, x\, \mathsf{M})\, \mathsf{A}))\, \mathsf{P}$$

yields a term that β-reduces to:

$$\exists x.(\mathsf{article}\, x) \wedge (\mathsf{wrote}\, \mathbf{m}\, x) \wedge (\mathsf{read}\, \mathbf{p}\, x)$$