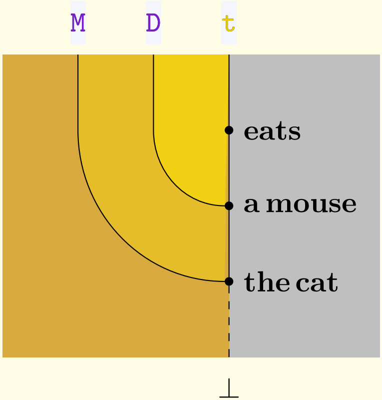
Formalizing Typing Rules for Natural Languages using Effects

Matthieu Boyer

18th April 2025 at CLaY

Plan

What the hell am I doing?



Defining definites and indefinites

A semantic theory for natural language is generally defined by three components:

- A syntax, for the generation of grammatical expressions, which we will not really discuss

Defining definites and indefinites

A semantic theory for natural language is generally defined by three components:

- A syntax, for the generation of grammatical expressions, which we will not really discuss
- A lexicon, for defining the meanings of individual expressions, which we will discuss by necessity

Defining definites and indefinites

A semantic theory for natural language is generally defined by three components:

- A syntax, for the generation of grammatical expressions, which we will not really discuss
- A lexicon, for defining the meanings of individual expressions, which we will discuss by necessity
- A theory of composition, describing how the meanings of complex expressions are built from their parts.

Defining definites and indefinites

A semantic theory for natural language is generally defined by three components:

- A syntax, for the generation of grammatical expressions, which we will not really discuss
- A lexicon, for defining the meanings of individual expressions, which we will discuss by necessity
- A theory of composition, describing how the meanings of complex expressions are built from their parts.

We will call the pair (syntax, lexicon) the language.

Defining definites and indefinites

However this generates new difficulties:

(1) Every planet shines.

(2) A planet shines.

(3) No planet shines.

This means in particular that **Every**, **A** and **No** should all have the same type `e` since **shines** has type `e` $\rightarrow$ `t`.

Defining definites and indefinites

However this generates new difficulties:

(4) Every planet shines.

(5) A planet shines.

(6) No planet shines.

This means in particular that **Every**, **A** and **No** should all have the same type $\mathtt{e}$ since **shines** has type $\mathtt{e} \to \mathtt{t}$. We will in this presentation provide a computable extension of usual theories of composition, based on a type and effect system to avoid this issue, based on [**?**].

Plan

Plan

Slide 42

Definition 2.1 — Category. A (small) *category* is described by the following data:

0 A class of objects (nodes of a graph).

Slide 42

Definition 2.2 — Category. A (small) *category* is described by the following data:

0 A class of objects (nodes of a graph).

1 For a pair A, B of objects, a set $\mathrm{Hom}(A, B)$ of functions from A to B called *morphisms*, *maps* or *arrows*. We denote it by $f : A \to B$ or $A \xrightarrow{f} B$.

Slide 42

Definition 2.3 — Category. A (small) *category* is described by the following data:

0 A class of objects (nodes of a graph).

1 For a pair A, B of objects, a set $\mathrm{Hom}(A, B)$ of functions from A to B called *morphisms*, *maps* or *arrows*. We denote it by $f : A \to B$ or $A \xrightarrow{f} B$.

2 For all triplets A, B, C, a composition law $\circ_{A,B,C}$:

$$\begin{array}{rcl} \mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) & \to & \mathrm{Hom}(A, C) \\ (g, f) & \mapsto & g \circ f \end{array}$$

Slide 42

Definition 2.4 — Category. A (small) *category* is described by the following data:

0 A class of objects (nodes of a graph).

1 For a pair A, B of objects, a set $\mathrm{Hom}(A, B)$ of functions from A to B called *morphisms*, *maps* or *arrows*. We denote it by $f : A \to B$ or $A \xrightarrow{f} B$.

2 For all triplets A, B, C, a composition law $\circ_{A,B,C}$:

$$\begin{array}{rcl} \mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) & \to & \mathrm{Hom}(A, C) \\ (g, f) & \mapsto & g \circ f \end{array}$$

2 For all objects A, an identity map $\mathrm{id}_A \in \mathrm{Hom}(A, A)$.

Slide 42

Definition 2.5 — Category. A (small) *category* is described by the following data:

3 Associativity: $f \circ (g \circ h) = (f \circ g) \circ h = f \circ g \circ h$

Slide 42

Definition 2.6 — Category. A (small) *category* is described by the following data:

3 Associativity: $f \circ (g \circ h) = (f \circ g) \circ h = f \circ g \circ h$

3 Unitarity: $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$.

Slide 42

A few examples of categories are:

Set whose objects are Sets and arrows are functions between sets.

Slide 42

A few examples of categories are:

Set whose objects are Sets and arrows are functions between sets.

Top whose objects are Topological Spaces and arrows are continuous maps between those.

Slide 42

A few examples of categories are:

Set whose objects are Sets and arrows are functions between sets.

Top whose objects are Topological Spaces and arrows are continuous maps between those.

Grp whose objects are Groups and arrows are Group Homomorphisms.

Slide 42

A few examples of categories are:

- **Set** whose objects are Sets and arrows are functions between sets.
- **Top** whose objects are Topological Spaces and arrows are continuous maps between those.
- **Grp** whose objects are Groups and arrows are Group Homomorphisms.
- **Vec** whose objects are Vector Spaces on a field k and arrows are Linear Maps.

Commuter Rail, basically

The right language for categories is the commutative diagram one.

The associativity rewrites as:

$$
\begin{tikzcd}
 & B \arrow[r, "g"] & C \arrow[dr, "h"] & \\
A \arrow[ur, "f"] \arrow[urr, "g\circ f"'] \arrow[rrr, "h\circ(g\circ f)"'] & & & D
\end{tikzcd}
$$

Commuter Rail, basically

In Set we have the following commutative diagram:

$$\begin{tikzcd}
\mathbb{R} \arrow[r, "\times 2"] \arrow[d, ".^2"'] & \mathbb{R} \arrow[d, ".^2"] \\
\mathbb{R}^+ \arrow[r, "\times 4"] & \mathbb{R}^+
\end{tikzcd}$$

Commuter Rail, basically

In Set we have the following commutative diagram:

$$\begin{tikzcd} \mathbb{R} \arrow[r, "\times 2"] \arrow[d, ".^2"'] & \mathbb{R} \arrow[d, ".^2"] \\ \mathbb{R}^+ \arrow[r, "\times 4"] & \mathbb{R}^+ \end{tikzcd}$$

It simply states that:

$$\forall x \in \mathbb{R}, \left(2x^2\right) = 4x^2$$

Of wolf, and man.

Definition 2.7 Let $\mathcal{A}, \mathcal{B}$ be two categories. A functor $\mathcal{F} : \mathcal{A} \to \mathcal{B}$ is:

0 An object $F(A) \in \mathcal{B}$ for each object A of $\mathcal{A}$.

1 For each pair $A_1, A_2 \in \mathcal{A}$, a function:

$$\begin{array}{rrcl} F_{A_1,A_2}: & \mathrm{Hom}_{\mathcal{A}}(A_1, A_2) & \to & \mathrm{Hom}_{\mathcal{B}}(FA_1, FA_2) \\ & f & \mapsto & F(f) \end{array}$$

Of wolf, and man.

We ask the following equations to be satisfied:

$F(g \circ f) = F(g) \circ F(f)$, that is

$$
\begin{tikzcd}
A \arrow[r, "g"] \arrow[d, "F"] & B \arrow[r, "f"] \arrow[d, "F"] & C \arrow[d, "F"] \\
FA \arrow[r, "Fg"'] & FB \arrow[r, "Ff"'] & FC
\end{tikzcd}
$$

and $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$

O.K. Corral

Definition 2.8 A natural transformation is a functor in the category of small categories and functors. If $F, G : \mathcal{A} \Rightarrow \mathcal{B}$ are functors, a natural transformation θ from F to G is, for each object of $\mathcal{A}$ a function θ_A such that the following diagram commutes for all $f : A \to B$.

$$\begin{tikzcd} FA \arrow[r, "Ff"] \arrow[d, "\theta_A"'] & FB \arrow[d, "\theta_B"] \\ GA \arrow[r, "Gf"] & GB \end{tikzcd}$$

O.K. Corral

Definition 2.9 An adjunction $L \dashv R$ between two functors $L : \mathcal{A} \to \mathcal{B}$ and $R : \mathcal{B} \to \mathcal{A}$ is a pair of natural transformations $\eta : \mathrm{Id}_{\mathcal{A}} \Rightarrow R \circ L$ and $\varepsilon : L \circ R \Rightarrow \mathrm{Id}_{\mathcal{B}}$ verifying the zigzag equations:

$$\begin{tikzcd}
\mathcal{A} \arrow[r, "\mathrm{id}_{\mathcal{A}}"] \arrow[dr, "L"'] \arrow[dr, phantom, shift left=4, "\boxed{\eta}"] & \mathcal{A} \arrow[dr, "L"] \arrow[dr, phantom, shift right=4, "\varepsilon"] & \\
 & \mathcal{B} \arrow[u, "R"] \arrow[r, "\mathrm{id}_{B}"'] & \mathcal{B}
\end{tikzcd}
=
\begin{tikzcd}
\mathcal{A} \arrow[rr, bend left, "L"] \arrow[rr, bend right, "L"'] \arrow[rr, phantom, "\Downarrow \mathrm{id}_L"] & & \mathcal{B}
\end{tikzcd}$$

Plan

Yu-Gi-Oh

Definition 2.10 A product of two objects A and B in a category φ is a triplet

$$(A \times B, \pi_1 : A \times B \to A, \pi_2 : A \times B \to B)$$

$$\begin{tikzcd} A & & B \\ & A \times B \arrow[ul, "\pi_1"] \arrow[ur, "\pi_2"'] & \end{tikzcd}$$

such that for all pair of arrows $X \xrightarrow{f} A$ et $X \xrightarrow{g} B$, there is a unique $h : X \to A \times B$ such that $f = \pi_1 \circ h, g = \pi_2 \circ h$.

Magic

Definition 2.11 A terminal object $\mathbb{1}$ in a category $\Gamma \vdash$ is an object such that for all A in $\Gamma \vdash$ there is one and only one arrow $A \to \mathbb{1}$.

An initial object is the same thing with the arrows reversed.

Definition 2.12 A category is cartesian if all products exist and it has a terminal object $\mathbb{1}$.

Cartesian products and terminal objects are unique, up to isomorphism.

Go Fish

Definition 2.13 A cartesian closed category is a cartesian category where we define for each object A a functor $A \Rightarrow \Gamma \vdash\!\!\to \Gamma \vdash$ right adjunct to $A \times \Gamma \vdash\!\!\to \Gamma \vdash$. This means we have bijections $\Phi_{X,Y} : \mathrm{Hom}\left(A \times X, Y\right) \to \mathrm{Hom}\left(X, A \Rightarrow Y\right)$ called currification bijections..

Go Fish

Definition 2.14 A cartesian closed category is a cartesian category where we define for each object A a functor $A \Rightarrow \Gamma \vdash\to \Gamma \vdash$ right adjunct to $A \times \Gamma \vdash\to \Gamma \vdash$. This means we have bijections $\Phi_{X,Y} : \mathrm{Hom}\left(A \times X, Y\right) \to \mathrm{Hom}\left(X, A \Rightarrow Y\right)$ called currification bijections..

Set is a cartesian closed category, where currification is defined by the partial application of high arity functions to a subset of their arguments.

Use the Types, Luke

Definition 2.15 The λ-calculus is defined by the following basic grammar:

$$\begin{array}{rcll} E & ::= & x \in Var & \textit{(Variables)} \\ & | & App(E, E) & \textit{(Application)} \\ & | & \lambda x.E & \textit{(Evaluation)} \end{array}$$

Use the Types, Luke

Definition 2.16 We give ourselves a set of type variables $TyVar$ and define types by:

$$\begin{array}{rcl} A, B & ::= & \alpha \in TyVar \\ & \mid & A \times B \\ & \mid & \mathbb{1} \\ & \mid & A \Rightarrow B \end{array}$$

A context $M = x_1 : A_1, \ldots, x_n : A_n$ is a list of pairs $x_i : A_i$ with a variable x_i and a type A_i, all the variables being different.

Fudge Supreme

Definition 2.17 A typing $\Gamma \vdash M : A$ is a triplet composed of a context Γ, a λ-term M and a type A, such that all free variables of M are in Γ. A proof tree for a typing judgement is constructed inductively from a set of rules of the form:

$$\frac{}{x : A \vdash x : A}\ \text{Var}$$

$$\frac{\Gamma \vdash M : A \Rightarrow B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash App(M, N) : B}\ \text{App}$$

Abstract Concrete

We will place ourselves in the functional programming paradigm, where everything is a function. Let $(\mathcal{C}, \times, \perp, \Rightarrow)$ be a cartesian closed category. The type variables are the object of our category.

Abstract Concrete

We will place ourselves in the functional programming paradigm, where everything is a function. Let $(\mathcal{C}, \times, \perp, \Rightarrow)$ be a cartesian closed category. The type variables are the object of our category.The type of an arrow $A \xrightarrow{f} B$ is $A \Rightarrow B \in \mathcal{C}$. For constants of type A in our language, we can also say their type is $\perp \to A$ to keep the function interpretation.

Plan

Plan

Underfull hbox

A side-effect is something that results from a computation without it being its actual return value. It could be:

Underfull hbox

A side-effect is something that results from a computation without it being its actual return value. It could be:

- The printing of a value to the console;

Underfull hbox

A side-effect is something that results from a computation without it being its actual return value. It could be:

- The printing of a value to the console;
- The modification of a variable in memory;

Underfull hbox

A side-effect is something that results from a computation without it being its actual return value. It could be:

- The printing of a value to the console;
- The modification of a variable in memory;
- Non-determinism in the result.

Underfull hbox

A side-effect is something that results from a computation without it being its actual return value. It could be:

- The printing of a value to the console;
- The modification of a variable in memory;
- Non-determinism in the result.

A way to model effects when in a typing category is by using functors [**?**].

Liberty

Let $\mathcal{L}$ be our language:

- $\mathcal{O}(\mathcal{L})$ is the set of words in the language whose semantic representation is a function.
- $\mathcal{F}(\mathcal{L})$ the set of words whose semantic representation is a functor.

Let $\mathcal{C}$ be a cartesian closed category adapted for our language.

Liberty

Let $\mathcal{L}$ be our language:

- $\mathcal{O}(\mathcal{L})$ is the set of words in the language whose semantic representation is a function.
- $\mathcal{F}(\mathcal{L})$ the set of words whose semantic representation is a functor.

Let $\mathcal{C}$ be a cartesian closed category adapted for our language.
Let $\bar{\mathcal{C}}$ the free categorical closure of $\mathcal{F}(\mathcal{L})(\mathcal{O}(\mathcal{L}))$.
Then our types are defined as: $\star = \mathrm{Obj}\left(\bar{\mathcal{C}}\right)/\mathcal{F}(\mathcal{L})$.
Let $\star_0 = \mathrm{Obj}(\mathcal{C})$.
We get a subtyping relationship based on the procedure used to construct $\bar{\mathcal{C}}$.