Algebraic Models for Linguistics
Inspired by Theoretical Physics

Matilde Marcolli (Caltech)

Run Run Shaw Lecture
Stony Brook, March 27, 2024
Based on joint work with Noam Chomsky and Robert Berwick


To appear as our forthcoming monograph “Mathematical structure of syntactic Merge”, MIT Press
What is Linguistics?

- **Linguistics** is the scientific study of language
- What is Language? (langage, lenguaje, ...)
- What is a Language? (lange, lengua,...)

Similar to ‘What is Life?’ or ‘What is an organism?’ in biology

- *natural* language
  as opposed to artificial (formal, programming, ...) languages
- language (in human brains) is a complex and highly structured phenomenon: modeling production, acquisition, parsing; similarities and differences with natural language handling by AIs, etc

- The point of view we will focus on:
  Language is a kind of *Structure*

What does it mean to model? what does it mean to explain?
fundamental question about the nature and purpose of science
Why mathematics?

- mathematics is the study of *structures*
- but isn’t it about *numbers*? no... numbers matter in mathematics only because they have interesting structures
- ... *and so does language*

Most important research problems in mathematics revolve around understanding different kinds of structures that may have to do with numbers, with geometric objects, or with more abstract algebraic properties.

Structures understood by mathematics played a fundamental role in the development of *theoretical physics* in the 20th century as well as today... can linguistics be studied with the same principles and methods as theoretical physics?
Example: numbers as structures

- **prime numbers** are integer numbers that are only divisible by themselves and 1

- they *generate multiplicatively* the integers (as structure building operation)

- many mysteries and deep open problems regarding the structure of prime numbers (e.g. the Riemann hypothesis)
What about language?

language exists at many different levels of structure just as physics looks very different at different scales:

- units of sound (phonology)
- words (morphology)
- sentences (syntax)

Syntax is the large scale structure of language, it is robust and highly structured, it is crucial for the compositionality of language, and necessary to the encoding of complex meaning.
How is language “a structure”?  

Example: look at this sentence

*I shot an elephant in my pajamas – what it was doing in my pajamas, I’ll never know* (Groucho Mark)

...why is it funny? because it conflates two different structures

...why is it funny? because it conflates two different structures

A sentence is **not** just a string of words!

We perceive *structural relations* not *proximity relations* in the ordering of words: structure reflects the different possible ways sentences are generated.
Generative Linguistics

... Chomsky describes it as the study of language as a structure of discrete infinity

Why discrete (countable) infinity?

- Sentence formation in language allows for recursions.
- For example, you can “help someone” and you can “help someone help someone” and you can “help someone help someone help someone...”
- In practice we would not use arbitrarily long iterations but our brain immediately understands that such a recursion could continue:
  1. could continue
  2. would remain grammatically correct
  3. would remain meaningful
- Recursions are the telltale sign of a computational process.
Generative Linguistics – The Questions:

- what is the fundamental *generative, computational* principle that allows for the richness of human language and the infinite possibility of sentences?
- what are the *constraints and rules* of this generative process?
- what principles *structure and organize* the diversity of linguistic forms we see across human languages?
- how does this structure formation and structure recognition happen in the human brain?
- how is language acquired?
- how did the faculty of language evolve?
- how do languages change dynamically over time?
Generative Linguistics – The History
the field was initiated and developed by Noam Chomsky

Formal languages, transformational grammar (1950s-1970s):

- N. Chomsky, *Syntactic structures*, Mouton 1957

Principles and parameters, Government and Binding (1980s):


Minimalism (1990s):


New Minimalism: Merge and SMT (starting ∼2000, then 2013-now):

The process of structure formation

- what language appears to look like

  0022010021112000121220000200211\ldots

- what language actually looks like

free symmetric Merge: the key structure formation operation
Syntax as a computational process

- *simple* core computational structure: Merge

This process generates structures
- called *hierarchical structures* (or *syntactic objects*) in linguistics
- called non-planar (or abstract) binary rooted trees by mathematicians

lexical items (words) at the bottom of the tree combine into a hierarchical structure

think of the tree dangling from the root, not lying in the plane: so the words at the bottom are *not* ordered sequentially
- a computational process of *structure formation*

  

  ![Diagram](image)

  - similar computational structures are natural in the context of fundamental physics (Feynman diagrams)
  - this generative process is a basic algebraic structure (known as a "free magma")
Externalization

- follows the generative process of structure formation (where sentences are hierarchical non-planar tree-structures)
- language is externalized through the sensory-motor system (speech, sign) as temporally ordered sequence of words
- equivalently: the trees acquire a *planar structure*
- language-dependent forms of *word order* + other rules
- example: English is head-initial, Japanese is head-final

$$
\begin{align*}
\text{VP} &= \text{verb phrase, TP} = \text{tense phrase, DP} = \text{determiner phrase} \\
\text{syntactic variability, syntactic parameters, geometry of syntax} \\
\text{(which constraints?)}
\end{align*}
$$
Summarizing the picture so far:

- Language starts as hierarchical structures: free floating (non-planar) binary rooted trees encoding only structural relations between words.
- The physical constraints of externalization (sound, motion) force a reduction to an ordered sequence.
- Our brain in language acquisition quickly learns from a small collection of such examples the (non-visible and complex) structure of syntax.

Note: the enormous computational difficulty of solving this inverse problem.... yet, our brain does it *easily*.
A closer look at structure formation (movement/transformation)

- we not only form sentences: we modify them, transform them, manipulate them according to precise rules (forming questions, changing to passive voice, etc)

- the Merge operation of structure formation has two aspects: Internal Merge and External Merge

- Internal Merge (responsible for “movement”) needs to reach inside formed structures (syntactic objects) for constituent parts (accessible terms) available for further calculation

- also structure formation is a “bottom-up operation” that proceeds in steps: intermediate scratchpad for operations (workspaces)
• workspaces

• accessible terms and syntactic objects
Algebraic structure appears: reinterpretation of Chomsky’s free Merge model in mathematical terms

- operations of assembling/disassembling
- applied to a class of combinatorial objects
- formalized by the notion of Hopf algebra
- same kind of algebraic structure well known to physics (Feynman graphs)
- provides a way to extract meaningful physical values from otherwise divergent Feynman integrals
- structure formation and parsing for meaningful interpretation accounted for by the same algebraic properties
Key idea: What is a Hopf algebra? assemble/disassemble operations (product/coproduct) with compatibility constraints

assembling: product operation

all possible ways of combining building blocks together
disassembling: coproduct operation

all possible ways of decomposing into two constituent pieces

Warning: this picture is deceiving as it makes it look like decomposition just undoes the composition operation but one can be richer than the other and still be compatible
Workspaces and product/coproduct

- **product** assembles and combines workspaces from lexical items and syntactic objects.

- A unique way of assembling.
- coproduct $\Delta$ disassembles in more interesting way accessing all substructures

multiple ways of disassembling corresponding to all different substructures (accessible terms)
Merge structure formation operation: acts on a workspace, produces a new workspace

1. **coproduct** produces the list of all accessible terms available form computation by disassembling the workspace

2. **grafting** together of two substructure (External Merge if two components; Internal Merge is a component and a substructure – movement/transformation)

3. **product** assembles back the new workspace

4. repeated applications of this sequence of operation starts from a set of words and builds (bottom-up) a sentence and its transformations
Example:

- extraction of substructures

\[ \Delta(\text{eaten } \sqcup \text{the } \text{apple}) = \text{eaten } \sqcup \text{the } \text{apple} \otimes 1 + \text{etc} \]

- formation of new structure (External Merge)

\[ \text{eaten } \sqcup \text{the } \text{apple} \rightarrow \text{eaten } \text{the } \text{apple} \]

- movement (Internal Merge)

\[ \text{was } \text{eaten } \text{the } \text{apple} \rightarrow \text{the } \text{apple} \text{was } \text{eaten } \text{the } \text{apple} \]
A closer look (slightly more technical for a few minutes)

the main ingredients so far:

- a set of lexical items $SO_0$
- a set of syntactic objects $SO = \mathcal{T}_{SO_0}$ binary rooted trees $T$
  with lexical items at the leaves
- set of workspaces $\mathcal{F}_{SO_0}$: forests $F$ whose components are syntactic objects
- this is just summarizing the picture

- **product** (assembly) operation $\sqcup$ puts together components into a workspace (e.g. the three components above $F = T_1 \sqcup T_2 \sqcup T_3$)
the main ingredients so far:

- **accessible terms** $v$ non-root vertex of syntactic object $T$ and subtree $T_v$ rooted at $v$

$$\text{Acc}(T) = \{ T_v \mid v \text{ non-root vertex of } T \}$$

- an accessible term can be extracted by a cut on an edge (above $v$); multiple accessible terms can be extracted by an admissible cut (no two cuts on same path from root to a leaf):
- **coproduct** \( \Delta \): all possible extraction of accessible terms via admissible cuts

\[
\Delta(T) = T \otimes 1 + 1 \otimes T + \sum_{v} F_v \otimes T/F_v
\]

\( F_v = T_{v_1} \sqcup \cdots \sqcup T_{v_n} \) accessible terms extracted by a cut, and \( T/F_v = \) what remains; first two terms include case where extract everything or nothing

- The *sum* is a way to list all possible decompositions simultaneously: *trick* to use such formal sums: vector space \( \mathcal{V}(\mathfrak{F}_{SO_0}) \) on the set of workspaces \( \mathfrak{F}_{SO_0} \) so coproduct is a map

\[
\Delta : \mathcal{V}(\mathfrak{F}_{SO_0}) \rightarrow \mathcal{V}(\mathfrak{F}_{SO_0}) \otimes \mathcal{V}(\mathfrak{F}_{SO_0})
\]

left and right channels of output are the two parts of the decomposition

- product goes the opposite way (two inputs assembled into one output)

\[
\sqcup : \mathcal{V}(\mathfrak{F}_{SO_0}) \otimes \mathcal{V}(\mathfrak{F}_{SO_0}) \rightarrow \mathcal{V}(\mathfrak{F}_{SO_0})
\]
- **grafting**: attach different components $F = T_1 \sqcup \cdots \sqcup T_n$ of a forest to a single root

$$
\mathcal{B} : T_1 \sqcup \cdots \sqcup T_n \mapsto T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_n
$$

- **Merge action**:

$$
\mathcal{M}_{S,S'} = \sqcup \circ (\mathcal{B} \otimes 1) \circ \delta_{S,S'} \circ \Delta
$$

1. coproduct extracts accessible terms
2. $\delta_{S,S'}$ selects matching terms
3. grafting forms new structure (Merge)
4. product reassembles new workspace
Algebraic properties of Hopf algebras

- **coassociativity**: good behavior under iteration
  \[(\text{id} \otimes \Delta) \circ \Delta = (\Delta \otimes \text{id}) \circ \Delta\]

- **product/coproduct compatibility**: not inverse operations but compatible
  \[\Delta \circ \sqcup = (\sqcup \otimes \sqcup) \circ \tau \circ (\Delta \otimes \Delta)\]
  \[\tau\] reorders so left and right channels grouped together

- **antipode**: additional constraint relating product and coproduct (but follows from coproduct in this kind of Hopf algebra)

These properties impose **very tight constraints**: several aspects of the linguistic model, developed for empirical reasons, are **determined** by the algebraic constraints

*a lesson from physics*: let the algebra do the work for you of constraining the model, a lot of properties that seem imposed **ad hoc** will turn out to be determined by structural necessity
Why is this formal algebraic description useful?

- **lesson from theoretical physics** (same algebraic structure with Feynman diagrams instead of syntactic objects):
  - coproduct decomposition allows for extraction of meaningful physical values of Feynman integrals (consistently over substructures):

\[
\Delta \left( \begin{array}{c}
\text{\includegraphics[width=2cm]{feynman_diagram1}}
\end{array} \right) = \begin{array}{c}
\text{\includegraphics[width=2cm]{feynman_diagram2}} \otimes \mathbb{1} + \mathbb{1} \otimes \begin{array}{c}
\text{\includegraphics[width=2cm]{feynman_diagram3}}
\end{array} + 2 \begin{array}{c}
\text{\includegraphics[width=2cm]{feynman_diagram4}} \otimes \begin{array}{c}
\text{\includegraphics[width=2cm]{feynman_diagram5}}
\end{array}
\end{array}
\end{array}
\]

- a similar coproduct/grafting/product operation describes the recursive solution of the quantum equations of motion (Dyson–Schwinger equations)

\[
\begin{array}{c}
\text{\includegraphics[width=4cm]{feynman_diagram6}}
\end{array} = \frac{1}{3} \begin{array}{c}
\text{\includegraphics[width=4cm]{feynman_diagram7}} + \text{\includegraphics[width=4cm]{feynman_diagram8}} + \text{\includegraphics[width=4cm]{feynman_diagram9}}
\end{array}
\]

- **renormalization problem of quantum field theory**
Overall picture:
- In a similar way, formulation of the generative process of syntax in terms of coproduct/grafting/product operation allows for consistent mapping of syntactic structures and substructures to semantic spaces.
- This is called the syntax-semantics interface.
- Many different models of semantics but simplest is keeping track of proximity relations, agreement, similarity, co-occurrence (e.g., vector space models).
Syntax-Semantics interface: conceptual requirements

(not all would agree, but we take these as our background assumptions)

1. Autonomy of syntax
2. Syntax supports semantic interpretation
3. Semantic interpretation is, to a large extent, independent of externalization
4. Compositionality

- autonomy of syntax: Merge computational generative process of syntax independent of semantics
- syntax-first view: syntax-semantic interface proceeds from syntax to semantics
- two channels: from core Merge mechanism to Conceptual-Intentional system (syntax-semantics interface) and to Sensory-Motor system (externalization)
- compositionality: consistency across syntactic sub-structures
(from Chomsky et al. “Merge & SMT”)
Geometry of semantic spaces

- language is a recent evolutionary step: requires a *small* modification: Merge as single evolutionary step (Berwick-Chomsky)
- everything else already evolved: semantic/conceptual spaces not specific to language (syntax specific to language)...
  e.g. conceptual manifolds in vision
- basic structures for a semantic space
  1. proximity measurement (topology/metric)
  2. interpolation (convexity)
  3. agreement/disagreement measurements
Semantic parsing of syntactic structures

- lexical items map to semantic space \( s : SO_0 \rightarrow S \) (words in context)
- syntactic objects (trees) \( T \) also map to \( S \) using proximity and interpolation property of semantic space together with additional aspects (syntactic head, phases, labeling) on syntactic objects
Head and Phases

method to reduce combinatorial explosion of extraction of substructures, identifying essential substructures that are complete for semantic parsing.
The inverse problem

- syntax is computational generative structure, semantics is not
- ... but syntax casts a shadow of itself inside semantics through this recursive procedure of consistent semantic parsing of substructures
- **Question**: is this “static image” of syntax inside semantics, blurred by probabilistic data, enough to reconstruct the computational process of syntax?

This question is very relevant now in the context of language handling by Artificial Intelligence (large language models like Chat-GPT)
What is happening on the other side of the trenches?

- very large corpora of text produced by human language
- encodings of semantic relatedness (in the form of vectors)
- computational architecture: transformers, attention modules
- very large parallel computing

**Result** large language models (LLM): machines that produce (mostly) correct generation of sentences and language manipulation
What is being computed?

- start with string of words (lexical items, tokens) in a text
- to each assign 3 vectors (matrices): queries, keys, values
- these encode (statistically) other words that are structurally related to ("called by" or "calling for") the given word
- query and key vectors are paired (measure of relatedness) to generate probability weights
- these are used to weight and average values

Result: can complete tasks such as completing missing words from a sentence (BERT) or add a next word (GPT)
What is *actually* happening?

- conflicting results on handling of syntax by LLMs when syntactic structures become complex
- syntactic trees can be “seen” from the weights of attention modules (Mannings et al. 2020):

  - “poverty of the stimulus” for human learning versus “overwhelming richness of the stimulus” for LLM training
LLMs perform a (partial) solution of the inverse problem

- the keys and queries are a statistical proxy for the Generative Linguistics notion of syntactic relationship (c-command) and the corresponding positions (in terms of structural relations) in a syntactic tree
- very large parallel computing searching through huge corpora for an image of syntax projected upon semantics (a difficult and imperfect inverse problem)
- syntactic trees are imperfectly encoded in the weights of the attention modules and can be read from them

... a machine that makes plausible predictions (e.g. of the next word in a sentence): is this science?
physics as metaphor

- Quantum Field Theory: generative process of Feynman diagrams, assignment of meaningful physical values (renormalization) $\Rightarrow$ perturbative computation of Higgs boson production cross sections
Particle accelerators and detectors: solving an inverse problem that identifies inside enormous set of data traces of the correct diagrams/processes involving creation/decay of a Higgs particle through interactions of other particles sees “an image” of the QFT objects embedded into the set of data collected by detectors, against a noise background of a huge number of other simultaneous events
the generative process of syntax is embedded in LLMs in a conceptually similar way: its image is scattered in a probabilistic smear across large number of weights and vectors, trained over large data sets

signals of linguistic structures detectable against a background of probabilistic noise

LLMs do not “invalidate” generative syntax any more than particle detectors would “invalidate” Quantum Field Theory: quite the opposite

consequently:

LLMs are not a language theory, generative syntax is

LLMs are an experimental apparatus for the study of the inverse problem of the syntax-semantic interface

data and technology without theory do not constitute science

Where is the explanatory power? Where is the understanding?
The purpose of science is to obtain a concise conceptual explanation of natural phenomena, that should be testable, predictive, and essential (*entia non sunt multiplicanda praeter necessitatem*).

Predictions are needed for *falsifiability* of scientific theory, but are not the goal in themselves, the goal of science is *conceptual explanation*.

**Generative linguistics** aims at producing such explanations for the structure and functioning of language.

What is actually happening in LLMs *can* and *should* be understood by a careful mathematical modeling of what they compute and comparing it with mathematical models of generative syntax as produced by human brains:

- mathematics is a powerful explanatory tool, because it is both highly constrained and highly flexible;
- this is why it is the language of science (as Galileo said, the language in which the universe is written).
Thank You!