STRING DIAGRAMS FOR TEXT

VINCENT WANG-MAŚCIANICA

ST. CATHERINE'S COLLEGE
THE UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
2023

Contents

(Acknowledgements will go in a margin note here.)

$0\frac{1}{2}$ *Corrections*

There are several things I would like to get through in this long postscript, which has several purposes. In no particular order, the first is to settle most of the substantive corrections (recommended to me by my examiners Stefano Gogioso and Jules Hedges) in one go. The second is to situate this work with respect to formal linguistics as done by formal linguists. The third is to vent and find closure for myself.

$0\frac{1}{2}$.1 *Does it work?*

In the most important sense, no.

I will elaborate what I mean. There is a classical conception of structured approaches to AI/ML that permits capacities that go beyond what connectionist approaches are capable of, grounded in a kind of quasi-magical conception of mathematics as the ultimate form of understanding of — and hence control over — a phenomenon. It appears that part of the intellectual shock of generative AI is that one does not need to understand the mechanics of complicated phenomena in order to reliably induce them, given sufficient data and compute. This is obviously disappointing to many, and the highest sort of achievement that this thesis could have attained is to have been a source of hope. Even a year ago the prospects seemed grim, and part of what took me so long to write the introduction was to compromise the positive vision from some demonstration of the complete superiority of structured approaches (basically no longer tenable after the advent of ChatGPT with RLHF) to something weaker, such as a performant way to synthesise structural/symbolic and connectionist approaches that has some other kind of benefit, such as "interpretability" (whatever that means.)

So taking *"does it work?"* to mean *"does it justify the activity of mathematically formulating the structure of language with respect to non-instrinsic measures of value (such as practical application)?"*, the answer is no, modulo most measures of value that people would care about. I say this because we've done some experiments.

Dear Reader,

How are you? I am well, thank you.

Forgive me for writing in this informal register; it is easier for me than the academic style (which I am no good at), and I would like to get these corrections done as painlessly as possible. You see, in the year or so it has taken for me to get around to these corrections, a lot has happened. I have gotten married, and (thankfully relatedly) I am soon to be a father. So I've learnt that there are more important things in life than a thesis, and I have otherwise been busy drawing diagrams about other things. While we are on apologies, please forgive me also for the overall tone of this work, which editorialises, and vacillates between serious and light; in my defense, I am constitutionally a trifler [CITE] (or else I wouldn't have gotten into formal linguistics), and I am still in the process of finding my voice as a writer.

I should preface by reminding you that all of this is written long after the rest of the work, and also that any opinions expressed here are strictly my own. I will also try my best to keep personal opinions tucked away in the margins so that they may be safely ignored.

$0\frac{1}{2}$.1.1 *Have experiments been done?*

The approaches sketched out in Section **??** have been tried with neural networks in various ways by masters' students taking the Distributed Models of Meaning course offered in MFoCS here at Oxford, and separately it has been tried in the quantum setting **??** by talented scientists and engineers at the company at which I am employed at the time of writing. In the classical case, it works for some bAbI tasks, but at the cost of hardcoding the structure of the queries, and it doesn't outperform a transformer.

I could be wrong; it could be that all of the experiments done so far were in compute and data regimes that are too small to be indicative of how these approaches may scale. It could be that rather than the level of words and sentences there is some structural benefit to be obtained at the textual level, and so on, but I am not personally holding out hope any more. I do maintain that the symmetric monoidal (and hence string-diagrammatic) approach to formalising the structure of language is the best and most natural way to synthesise the mathematics with modern (practical) machine learning, so the fact that it doesn't work all that well leaves me with a very dim view of everything else. As a result of these experiments and other experiences, my own view of the role of structure has been further demoted. As far as natural language is concerned, at worst such structure appears to be some crutch for us humans: inductive biases that constrain and unnecessarily burden more sophisticated algorithms that can deal with the complexity of "the real world". At best, it appears to be tradeoff: computationally cheaper but worse answers.

$0\frac{1}{2}$.2 *Can you situate this work with respect to the literature in formal linguistics?*

One way to summarise things is that the technical contents of this thesis point towards a nearby counterfactual history where all the linguists in the "Garden of Eden period" [1] knew some of the modern structuralist mathematics that their programme obviously would have profited from. The fractiousness of linguists notwithstanding, it is my opinion that the interdisciplinarity of this thesis is accidental: from the perspective of the counterfactual history, the division of formal approaches to syntax and semantics (and some of the subdivisions thereof) would appear contrived. I am aware that this could come across as pretty arrogant and patronising talk, which is not my intention this time, because the point I am ultimately trying to make is that nothing in this thesis is particularly special, and could have been reinvented by anyone. Because I thought things through for myself for the most part, and since the relevant developments were due to collaborations with the similarly ignorant, text circuits and the other contents of this thesis owe no substantial intellectual debt to linguistics outside of perhaps Lambek and Firth, themselves far from the mainstream. Consequently, all of the parallels I am about to point out between the current stream of development and the main body of literature are not causally related. I take these parallels as indications of the "naturalness" of ideas that could have come about anytime and independently of specific individuals, and accordingly as evidence for the "nearbyness" of the counterfactual history I am trying to gesture at.

Before I go on to do the scholarly work of relating my stuff to the stuff of others, I just want to say: look, it's not in the culture of my people (mathy folk) to read . It's either uninteresting so we put the book down, or if it is interesting, we put the book down and try to rederive it for ourselves. Accordingly, if armchair introspection was good enough for Harris and Chomsky and so on, I figured it was good enough for me too. It's hard for me to care about any formal linguistics because I'm just not sure that telling mathematical stories about language is a meaningful activity anymore. To be fair, I think there is a lot of interesting *Linguistics* out there — in my mind the kind of distinction between formal versus respectable linguistics is exemplified by the anecdote of Daniel Everett uncovering the structure of Pirahã in the field, causing the formalist Chomsky to move the goalposts on what recursion means — but I have grown increasingly convinced that all of the effort to cast linguistics in mathematical terms up to now is best viewed as a kind of devotional or religious activity, a kind of benign way to kill time. Or perhaps it is a kind of crucible that refines the sensibilities of mathematical masochists. The only practical problem formal linguistics solves appears to be the gainful employment of overeducated fools such as myself.

[1]

$0\frac{1}{2}$.2.1 *On Pregroups to Text Circuits vs. Transformational Grammar to Formal Semantics*

This subsection about counterfactuals will also serve as an addendum to the cartoon literature review, which is folkloric, autochtonal, and maybe a little jingoistic: it is a caricature or myth of the field that we in it tell ourselves. I think that is sufficient for most readers, but if you are here, I owe you a critical and comparative retelling. In my view, the development of DisCoCat is only two minor counterfactuals removed from the lineage of mainstream formal semantics from Montague onto Heim & Kratzer and onwards. Moreover, both of these counterfactuals seem to rest only on the difference of when they began relative to the ambient development of mathematical formalisms and available computing.

First counterfactual: Truth-conditional vs. Vectorial Semantics

Montague semantics may be essentially characterised as the meeting of two ideas [CITE]: structure-preserving maps from syntax, and taking truth-conditions to be the essential data of semantics. On some accounts, only the former aspect of compositionality of semantics according to syntax is essential [CITE]. Accordingly, the first counterfactual is just the swapping of truth-conditional for vectorial semantics. Today there are several good reasons to prefer the latter over the former. First, the view that truth-conditions alone are the *sine qua non* of natural language meanings has been incompatible with correspondence theories of truth at least since Barr fixed Putnam's permutation argument [CITE, CITE]. Second, vectors as lists-of-numbers are more expressive and computationally practical, so much so in its current form that the very need for a formal account of "semantics" is put to question. Third, (feel free to disagree with me) with a rare few exceptions, the truth-conditional programme and its descendents are bankrupt, and worse, have terrible mathematical taste. A lot of mathematics has been marshalled [CITE,CITE,CITE] to salvage the programme by trying to force intensions and pragmatics and everything-in-the-world into the propositional mould, and it is unclear what all of this mathematics buys us except for more of the same. In practical terms, the increasing extent to which statistical language models adequately handle semantics exactly matches the decreasing extent to which a complicated mathematical account of the same is warranted, and consequently, in theoretical terms it seems unserious to assert that the study of the mathematical models themselves lends insight into the phenomena they are intended to be surrogates for; if that kind of truth-seeking were truly the aim, wouldn't modern formal linguists be lining up to pick apart large language models?

But all this criticism can only be said with the benefit of hindsight, and to give credit, it all must have seemed like a very good idea at the time. A model-theoretic, truth-conditional account of semantic data was the natural choice for a concrete target for the structure-preserving map, I speculate, for several reasons: Montague himself was a logician, and truth-conditions were at once flexible enough to capture (to a logician's satisfaction) some semantic phenomena of interest, while being amenable to computation *by hand*, as was necessarily the case at the time owing to the lack of computers. Certainly there was adequate sophistication manipulating vectors by that time as well, but the vectorial view would have been more difficult to calculate with, even restricted to a setting without nonlinearities. It could be argued that, in any case, vectorial semantics in the form of word-embeddings requires a degree of data-storage and computing faculties that would not have been available until fairly recently. Moreover, even the theoretical soil was arguably unready: category theory was insufficiently spread and understood, and hence a broadly accessible mathematical understanding of structure and structure-preservation outside of particular concrete instances was unavailable.

Second counterfactual: Generative vs. Typelogical views of syntax

In the origins of formal language theory taught in undergraduate computer science courses, a formal language is an acceptable subset of all possible strings from a stock of symbols. Context-free and context-sensitive grammars as rewrite systems are really generative in the everyday sense of the word, in that they are combinatoric abstract machines that produce acceptable strings. In the same way that pushdown automata parse context sensitive languages, typelogical grammars are another template for specifying grammars where parsing rather than production takes precedence; a proof-theoretic counterpart to combinatoric conceptions of grammar. Today "generative grammar" in some circles is just synonymous with "formal", which suggests a kind of unwarranted symmetry-breaking from path-dependency that could have also gone the other way if parsing formalisms such as typelogical grammars were more popular at the start. So in my personal mental model, there are (inter alia) two different kinds of mathematical formalisms for syntax, which are productive and parsing, and for any formal theory of syntax to gel properly with an analysis of communication, there's an onus on the modeller to provide a partnered formalism from the other side: for example, context free grammars as productive grammars get partnered with finite-state machines or, say, pregroup grammars that parse.

So the second counterfactual is just deciding to start from the parsing view rather than the productive one; to consider oneself a listener rather than a speaker. When it comes to natural language, perhaps the parsing and productive views would have been on more equal footing if computers were more advanced, because theory would have been held to account by the practical demands of serialising data structures as strings (production), and recovering them (parsing).

So if history had gone another way, quantum linguistics could have been the natural thing to do from the start, and it probably would not have required any detour through "quantum".

$0\frac{1}{2}$.2.2 *On Montague's conception of grammar*

This subsection about Montague's grammar was originally a postscript for Chapter **??** titled *A modern mathematician's guide to Montague's "Universal Grammar"*. I've moved it here so that it can live alongside other things formal linguists might be interested it. I originally wrote this out of spite, because some category theorist said (flippantly, in my mind) that Montague semantics was "just a so-and-so". It is a kind of in-joke and shibboleth among category theorists to say that "X is just a Y" where X is quite pedestrian and Y is some scary arcane nonsense. Anyway, I thought this guy was full of it and didn't do his homework, so I thought I would go and show him up by getting the news at the source and rubbing his face in it. Though I still don't believe he did his homework, I did mine, and embarrassingly it turned out he was more or less right.

In summary, to do "Montague semantics" means taking structure-respecting homomorphisms from grammar to meaning (**?**). Montague (likely) considered grammars to be coloured operads; Montague's "algebras" are (multi-sorted) clones, which are in bijection with (multi-sorted) Lawvere Theories, which are equivalently coloured operads, which may be viewed as special cases of coloured PROPs. Hence text circuits share a mathematical lineage with many other mathematical conceptions of grammar, while also enjoying Montague semantics.

Montague semantics/grammar as Montague envisioned it is largely contained in two papers – *Universal Grammar* (**?**), and *The Proper Treatment of Quantifiers in English* (**?**) – both written shortly before his mysterious death in 1971, so there were no opportunities for further elaboration. The methods employed were not mathematically novel – the lambda calculus had been around since the 1930s (**?**), and Tarski and Carnap had been developing intensional higher-order logics since the 40s (**?**) – but for linguists who, by-and-large, only knew first order predicate logic, these methods were a tour-de-force that solved longstanding problems in formal semantics.

There is a natural division of Montague's approach into two structural components. According to Partee — herself a formal semanticist, advocate, and torch-bearer for Montague — the chief interest of Montague's approach (as far as his contemporary linguists were concerned) lay in the following ideas (?):

1. Take truth conditions to be the essential data of semantics.
2. (a) Use lambdas to emulate the structure of syntax...
   (b) ...in a typed system of intensional predicate logic, such that composition is function application.

More precisely, Montague devised a higher-order intensional logic for the first point, and the notion of a structure-preserving map from syntax to semantics for the second. The truth-conditional perspective was important at the time for enabling semantic computation, but within formal semantics there arose other perspectives on the nature of formal semantics, such as inquisitive (?) and update semantics (?). Today, the empirical evidence we have from vector-based methods in computational linguistics is that none of those conceptions of semantics are intrinsically interesting or canonical: certainly none are procedurally necessary for a broad conception of practicality. So let's nevermind points 1 and 2b.

I have split the second point to highlight the role of lambdas. This element was the crux of the Montagovian revolution: according to Janssen in a personal communication with Partee from 1994, lambdas were "...the feature that made compositionality possible at all." Using lambdas to make the semantic domain compositional then gave a target for the structure-preserving homomorphism from the syntactic domain. Today, we have more refined ways to grant structure to semantic domains using category-theoretic tools. So let's redact "lambdas" from 2a.

What remains that is of interest is the question of what Montague considered the structure of syntax to be. This is worth understanding, since we claim text circuits are a "structure of syntax", and that functorial interpretation of text circuits in symmetric monoidal categories is Montagovian semantics in spirit if not in letter. So let's begin. In Section 1 of *Universal Grammar*, Montague's first paragraph establishes common notions of relation and function – the latter he calls *operation*, to distinguish the n-ary case from the unary case which he calls *function*. This is all done with ordinals indexing lists of elements of an arbitrary but fixed set A, which leads later on to nested indices and redundancy by repeated mention of A. We will try to avoid these issues going forward by eliding some data where there is no confusion, following common modern practice. Next, Montague introduces his notion of *algebra* and *homomorphism*. I will shunt the reproductions of the definitions to the margin. First he separates the data of the carrier set and the generators from the *polynomial operations* that generate the term algebra. All of Montague's algebras also come equipped with the data of *identities*, *constants*, and *composition*.

Section 2 seeks to define a broad conception of syntax, which he terms a *disambiguated language*. This is a free clone with carrier set A, generating operations F_γ indexed by $\gamma \in \Gamma$, along with extra decorating data:

Definition $0\frac{1}{2}$.2.1 (Generating data of an Algebra)**.** Let A be the carrier set, and F_γ be a set of functions $A^k \to A$ for some $k \in \mathbb{N}$, indexed by $\gamma \in \Gamma$. Denoted $\langle A, F_\gamma \rangle_{\gamma \in \Gamma}$

Definition $0\frac{1}{2}$.2.2 (Identities)**.** A family of operations populated, for all $n, m \in \mathbb{N}$, $n \leq m$, by an m-ary operation $I_{n,m}$, defined on all m-tuples as

$$I_{n,m}(a) = a_n$$

where a_n is the n^{th} entry of the m-tuple a.

Definition $0\frac{1}{2}$.2.3 (Constants)**.** For all elements of the carrier $x \in A$, and all $m \in \mathbb{N}$, a constant operation $C_{x,m}$ defined on all m-tuples a as:

$$C_{x,m}(a) = x$$

Definition $0\frac{1}{2}$.2.4 (Composition)**.** Given an n-ary operation G, and n instances of m-ary operations $H_{1 \leq i \leq n}$, define the composite $G(H_i)_{1 \leq i \leq n}$ to act on m-tuples a by:

$$G(H_i)_{1 \leq i \leq n}(a) = G(H_i(a))_{1 \leq i \leq n}$$

N.B. the m-tuple a is copied n times by the composition. Writing out the right hand side more explicitly:

$$G\Big(\, (\, H_1(a) \, , \, H_2(a) \, , \, \dots \, , \, H_n(a) \,) \, \Big)$$

Definition $0\frac{1}{2}$.2.5 (Polynomial Operations)**.** The polynomial operations over an algebra $\langle A, F_\gamma \rangle_{\gamma \in \Gamma}$ are defined to be smallest class K containing all $F_{\gamma \in \Gamma}$, identities, constants, closed under composition.

Definition $0\frac{1}{2}$.2.6 (Homomorphism of Algebras)**.** h is a homomorphism from $\langle A, F_\gamma \rangle_{\gamma \in \Gamma}$ *into* $\langle B, G_\gamma \rangle_{\gamma \in \Delta}$ when

1. $\Gamma = \Delta$ and for all γ, F_γ and G_γ agree in arity
2. $h : A \to B$
3. For all γ and lists of arguments $\langle \mathbf{A} \rangle$, $h(F_\gamma(\langle \mathbf{A} \rangle)) = G_\gamma(h(\langle \mathbf{A} \rangle))$

1. $(\delta\ \in)\Delta$ is an (indexing) set of syntactic categories (e.g. $\mathtt{NP}$, $\mathtt{V}$, etc.). Montague calls this the *set of category indices*. $X_\delta \subseteq A$ form the *basic expressions* of type δ in the language.

2. a set S assigns types among $\delta \in \Delta$ to the inputs and output of – not necessarily all – F_γ.

3. a special $\delta_0 \in \Delta$ is taken to be the type of declarative sentences.

This definition is already considerably progressive. Here are several observations:

(⋆) Because there is no condition of disjointness upon the X_δ — a view that permits the same word to play different syntactic roles — (1) permits the same basic expression $x\ \in\ A$ to participate in multiple types $X_\delta \subseteq A$.

Condition (2) misses being a normal typing system on several counts. There is no condition requiring all F_γ to be typed by S, and no condition restricting each F_γ to appear at most once. This raises the possibilities that:

(†) some operations F go untyped.

(‡) some are typed multiply.

Taking a disambiguated language $\mathfrak{U}$ on a carrier set A, Montague defines a *language* to be a pair $L\ :=\ \langle \mathfrak{U}, R\rangle$, where R is a relation from a subset of the carrier A to a set $\mathtt{PE}_L$, the set of *proper expressions* of the language L. It appears that a purpose of R defined in this way is to permit the modelling of syntactic ambiguity, where multiple elements of the term algebra $\mathfrak{U}$ (corresponding to syntactic derivations) are related to the same "proper language expression".

It appears that Montague's intent was to impose a system of types to constrain composition of operations, but the tools were not available for him. Montague addresses (†) obliquely, by defining $\mathtt{ME}_L$ to be the image in $\mathtt{PE}_L$ of R of just those expressions among A that are typed. Nothing appears to guard against (‡), which causes problems as Montague expresses structural constraints (in the modern view) in terms of constraints on the codomain of an interpreting functor (cf. Montague's notion of syntactic categories that "generate"). One consquence, in conjunction with (⋆), is that every multiply typed operation F induces a boolean algebra where the typings are the generators and the operations are elementwise in the inputs and output. Worse problems occur, as Montague's clone definition include all projectors, and when defined separately from the typing structure, these projectors may be typed in a way that permits operations that arbitrarily change types, which appears to defeat the purpose. I doubt these artefacts are intentional, so I will excerise my hermeneutical rights and assume his intent was a type-system with categorical semantics as we would use today. In so doing, we can summarise things fairly succinctly.

Proposition $0\frac{1}{2}$.2.7. Montague's grammars are coloured operads.

Proof. Definition $0\frac{1}{2}.2.2$ is equivalent to asking for all projections. Definitions $0\frac{1}{2}.2.2$ and $0\frac{1}{2}.2.4$ together characterise Montagovian algebras as (concrete) clones, which then generalised to (abstract) clones, which were then discovered to be in bijection with Lawvere theories (**?**); by an evident extension of [Prop 3.51] in that same paper to the typed case, a *disambiguated language* is a multi-sorted Lawvere theory without relations, where the sorts are generated from products of a pointed set $(\Delta\ ,\ \delta_0\ :\ 1 \rightarrow \Delta)$. Lawvere theories are themselves a special case of operadic composition (**?**). Operadic composition naturally viewed as that of coloured trees, which is equivalently depicted as nesting expressions according to the tree-structure. Interpreting colours as types, operadic composition subsumes whatever one might wish to do with a typed gentzen-style sequent system where rules are multi-input single-output. □

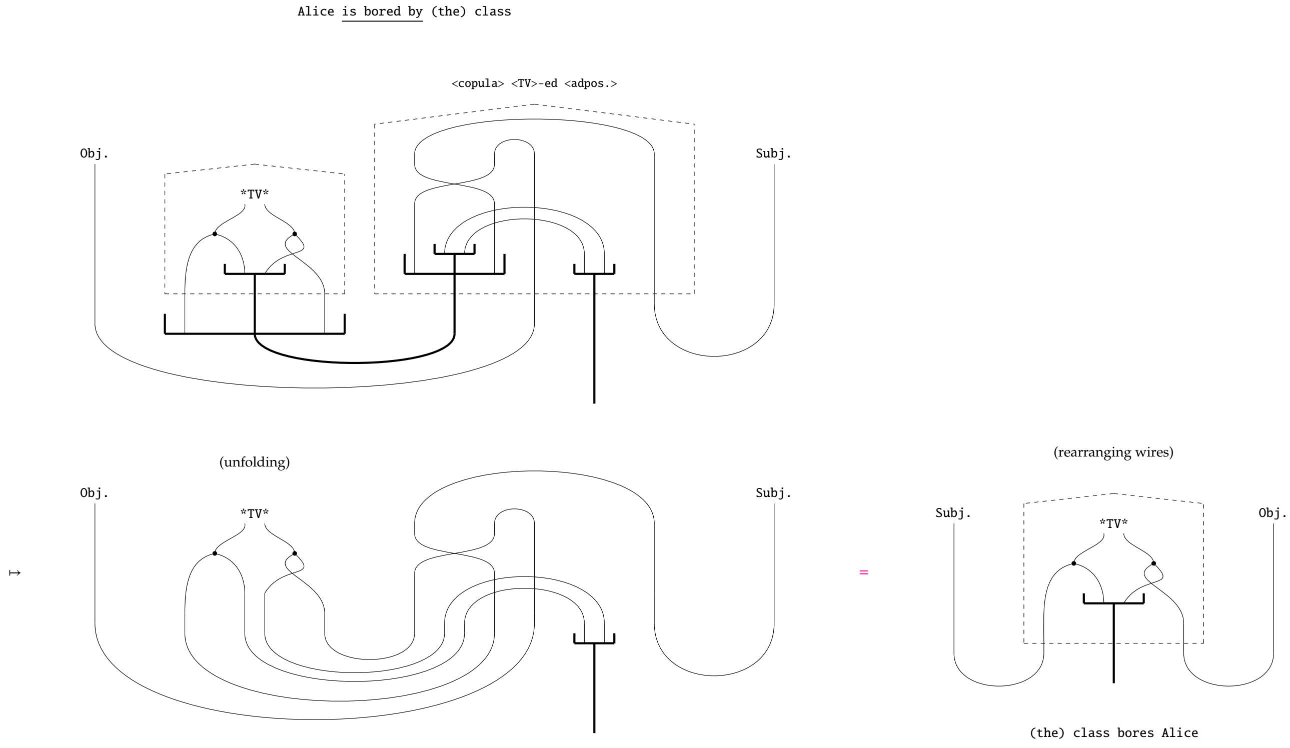
While typelogical grammars stop there, PROPs generalise operadic composition to multi-input multi-output, and as combinatorial specifications for string diagrams, weak n-categories generalise PROPs, and so we have shown weak n-categories to subsume a distinct evolutionary line of formal syntax from CFGs to TAGs.

$0\frac{1}{2}.2.3$ *On Deep Structure, the Universal Base Hypothesis, and the "Lexical Objection"*

As the story roughly goes [CITE], once upon a time some clever people dreamt up a *deep structure* that captured the syntactic relationships between the surface appearance of words in a sentence. Then they realised that by manipulating the deep structure with a *transformational grammar* (TG), they could relate sentences that seemed natural to relate. To be able to express the difference between related sentences as a single mathematical move was exciting stuff! Well, how far could one impose order on the jungle of syntax? Here's a thought: if language is perhaps something innate for humans (which as far as anyone knew was a reasonable assumption at the time), how about this for an ambitious idea: let's all try to find a single, canonical underlying mathematical structure that could be specialised to obtain a model for the syntax of any natural language: a *universal base* [CITE]. Of course, to do this, it was necessary to refine the mathematical models to more closely match the empirical data of English, so these clever people (who were now called formal linguists) got to work together, and that was exciting and fun. In those halcyon days, it was also believed that this could be done without a consideration of the meanings of words, because the assumption that grammar was somehow upstream of meanings, called *the autonomy of syntax* [CITE], which is also soft prerequisite assumption if one holds that semantics is compositional according to syntax. But as TG got better at capturing English, this belief about the autonomy of syntax started running into trouble. For instance, some of the marking data on generators required evidently semantic data, such as indication of whether or not something was a fish [CITE]; that is troubling, because why would the one true syntax care about whether something was a fish? So, some of the formal linguists decided to take semantics more seriously. But that meant that syntax alone wasn't going to cut it, and that meant throwing away some really nice ideas, like the autonomy of syntax, or (heaven forbid) the existence of a universal base that everyone was working hard to find. Some of the other formal linguists who really liked syntax didn't like the suggestion that syntax isn't special, and so they thought that taking semantics seriously was a silly idea and a bad move. And then everyone started saying mean things to each other, and then there was a big fight, and that was exciting and sad.

The fight is long settled, but I'm not sure that it's bedtime yet. My opinion is that the fight didn't really have to happen if they all just waited a while for computers to get better. I'm going to focus now on a particular example of how text circuits instantiated with vectorial semantics obtained from data might have patched things over, by sketching how one might address one of (in my opinion), the more deadly counterexamples to the autonomy of syntax, called the "lexical objection", which I'll explain now in grownup terms. One of the successes of TG was to demonstrate the essential sameness of sentences that look different on the surface, for example, a passive voice relationship like `Alice is bored by the class` and `The class bores Alice`.

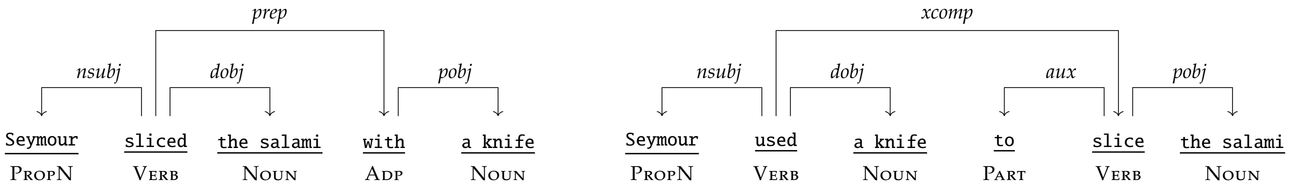
TG is pretty familiar, so I'll show how I handled this kind of grammar equation [CITE] via topological reasoning in pregroup diagrams: (move figures to side.)



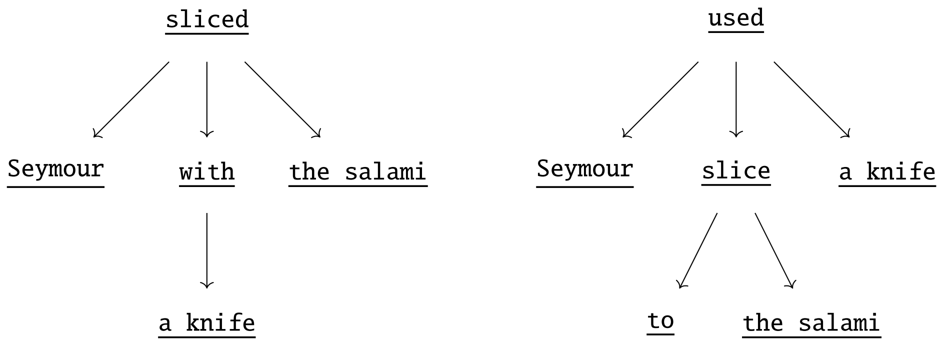
Whether it is TG or some other formal machinery that expresses it, if there is a universal base, then it probably ought to tell us about whether pairs of sentences of the kind above are really the same thing said differently, or else it wouldn't deserve such an important adjective like "universal". Now, if autonomy of syntax is true, then all of these grammar equations ought to be only concerned with the form or shape of the sentences involved, not on the particular words from your lexicon that are put in the place of subjects and verbs and ob-

rel pron circuit example for "that"

jects and what-have-you. If there were such a lexical dependency, then one would have to posit special rules for grammar that are dependent on the meanings of particular words, so syntax wouldn't be autonomous. The kicker is that there seem to be examples of such lexical dependencies, and that is the lexical objection. Here is one example now, in the form of two sentences about Seymour making lunch [CITE] along with their dependency grammar parses [CITE]:



These are two sentences that seem to mean the same thing, and if there is a universal base and autonomy of syntax is true and semantics is compositional, then it had better be that there's a purely syntactic explanation for the equivalence, just like the cases of passive voices and relative pronouns and so on. Before we continue I want to head off a class of objections to this line of argument that says that there are differences in the meaning of these two ways of talking about Seymour, because by construing "meaning" to be suitably encompassing, there are for instance emotive or topical differences in the two sentences in the context of, say, a story or a poem. So let's call this the poetic objection. Taken to its logical conclusion, the poetic objection concludes that any two sentences that aren't exactly equal are meaningfully distinct, modulo a suitably generous interpretation of what meaning is. This trivialises semantics, insofar as semantics is about constructing equivalence classes on syntax, so we can be done asking questions right away. Just on the basis of what is more difficult and interesting and potentially informative, the current line of argument is preferable because seeks to distinguish differences in ways meanings can be "the same", even if this notion of sameness is grounded in essentially nebulous and intuitive judgements, because we might at least flesh out what the boundaries of those judgements are and learn something in the process. So let's agree to carry on, unfolding the syntactic structures of the sentences into something familiar and treelike:



Now we have a problem: the root verbs are different, the shapes of the trees are completely different, and the leaf labels are juggled around. How would we transform one tree into the other? Whatever the method, it had better have something to do with the word `used`, because replacing `use` with some other word would change the meaning entirely. Fine, then let's try to save the autonomy of syntax by recategorising `use` to be a grammatical function word, in the same league as copulas and pronouns and adpositions. Then we would be able to have rewrites relating the two trees above by fiat, but what would the general rule look like? It seems like the necessary and sufficient conditions to permit such an equivalence would have to boil down to something to something akin to a frame [CITE], that tells us when we have an agent performing an action on a patient with a tool, that the same as an agent using a tool to perform an action on a patient:

$$\begin{cases} \textsc{Agent Action Patient } \texttt{[with/using]} \textsc{ Tool} \\ \textsc{Agent } \texttt{[uses]} \textsc{ Tool } \texttt{[to/for]} \textsc{ Action Patient} \end{cases}$$

Wait a minute, isn't that fundamentally about meanings as we humans experience it in the world? How could the notion of tools be part of syntax? If we were to take this observation seriously, as in try to formalise these kinds of equivalences founded on such frames, then we would be retreading the path of generative semantics, cognitive linguistics, all the way to expert systems in GOFAI. But all of that would require potentially abandoning universal bases (who's to say that such frames are not culturally dependent) and the autonomy of syntax (if meanings can affect grammar, then that's akin to breaking the central dogma of biology: suddenly there's very little hope to untangle the phenomena because it's a mess of everything affecting everything). So you can probably appreciate why some die-hard formal grammarians didn't like this whole idea and why there was a big fight. Ok, now here is a sketch proposal to resolve the problem. Let's first turn everything both sentences into text circuits, like so:

palceholder

Now we can interpret the frame as an equality condition on these two circuits, and if we instantiate all the boxes with, say, neural networks, we can express the equality condition as a loss function to be minimised [CITE](pattlang), such as:

bigeq

We would still have to go find all the frames or somehow farm them from data, and if we wanted to not only be able to verify equalities of meaning but also be able to efficiently produce paraphrases we would have to solve the problem of circuit tomography in terms of known building blocks [CITE], but these are technical rather than conceptual problems: this is a way to incorporate both kinds of paraphrase equalities, syntactic and semantic, in the same mathematical framework. Taken in conjunction with vectorial semantics for lexicons which are a scalable solution for word-meanings, we have a compromise: here is something that works with modern technologies, and you import all of your formal grammar and formal semantics. There will remain autonomy of syntax to an extent, in that semantics is still compositional and syntax-directed but in such a way that learnt representations will also correspond to other syntactic representations in a meaning-sensitive way. Perhaps most importantly for those who care, this way of doing things lets us take both semantics and syntax seriously, and it leaves open the possibility that there does exist a universal base; by essentially outsourcing the additional complications of meaning to vectorial semantics informed by big data, we have — albeit in a different way — once again isolated the question of syntactic form on its own terms.

$0\frac{1}{2}$.2.4 *On communication, and the mathematical infeasibility of the Autonomy of Syntax*

Unless it is somehow unreasonable to analyse communication in terms of a producer and a parser, a form of the autonomy of syntax assumption is at odds with a form of compositionality of semantics: one of the two has to give. The thrust of Section **??** is that, even outside of the lexical objection, syntax must be constrained by a purely compositional semantics because of the constraints imposed by communication. I think this may be a genuinely new contribution to formal linguistics of a particularly nice sort, because it only really uses ideas that were already lying around. There are two branches of the development of Formal Linguistics writ large that I will attempt to relate this insight to here. The first is formal language theory and the relationship between formal languages and the abstract machines that generate them. The second is semantics in generative grammar.

The idea of tying together production- and parsing- machines for formal languages is well known, and is the conceptual basis of the Chomsky hierarchy: more complicated languages require more complicated parsing machines. See Figures **??** and **??**.

But there is something missing in this picture, and that is semantics. The finite state machine in this example only does a weak form of parsing, indicating whether a string is acceptable or not. One obvious way to incorporate semantics on the parsing side here is to upgrade the finite state machine into a variant pushdown

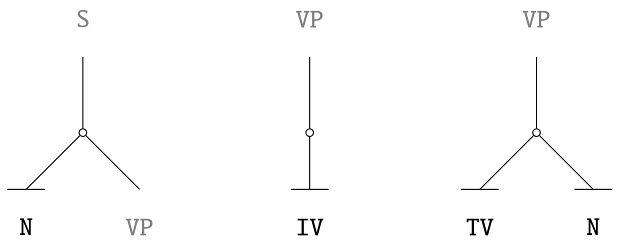


Figure 1: Here for example is a context free grammar for simple sentences that involve either a single transitive or intransitive verb. I've gathered together terminals, depicted as feet, into the basic generators.

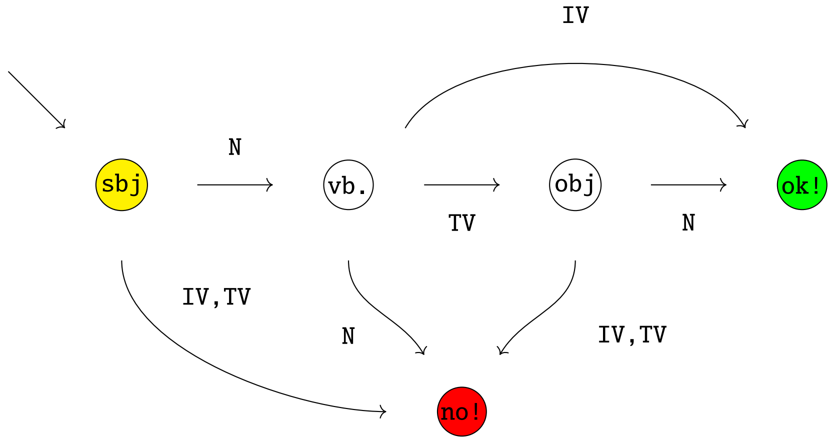


Figure 2: And here is its corresponding finite state machine. States are depicted as nodes, and transitions are depicted as labelled directed edges. This machine operates over the alphabet {N, IV, TV, and it only accepts those strings that are producible by its paired CFG.

automaton that simultaneously handles the transcription of the truth-conditional semantics in the λ-calculus. See Figure **??**.

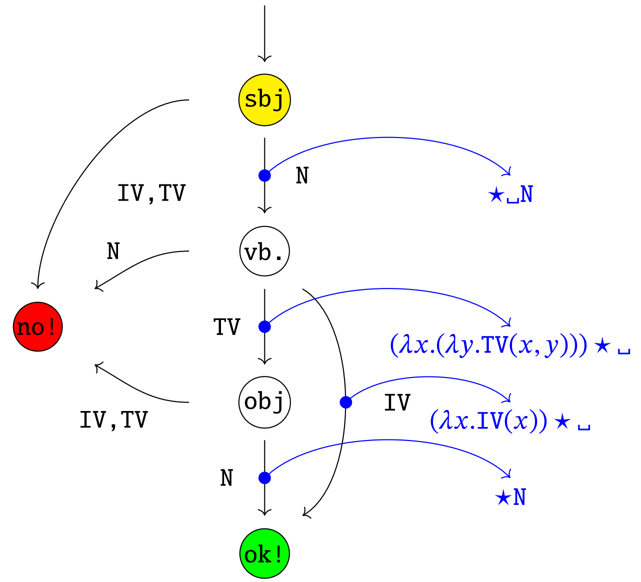


Figure 3: The blue arrows branching off of the transitions indicate what rewrites to perform on a side-buffer, where ★ denotes whatever was previously in the buffer, and ␣ indicates the placement of a cursor position where future rewrites are to take place. We're overloading notation here so that terminal symbols `N`, `IV`, `TV` in the buffer correspond to typed elements and functions $\texttt{N} \in D_e$, $\texttt{IV} \in D_{\langle e,t \rangle}$ and $\texttt{TV} \in D_{\langle e,\langle e,t \rangle\rangle}$ respectively, with respect to some bestiary of individuals/entities D_e and truth values D_t and their type closures in **Set** under exponentiation. An example stack-trace of the operation of this machine on the input `Ann loves Jan` is:

$$\varepsilon \xrightarrow{\texttt{Ann}} \text{␣}\ \texttt{Ann} \xrightarrow{\texttt{loves}} (\lambda x.(\lambda y.\texttt{loves}(x,y)))\ \texttt{Ann}\ \text{␣}$$

$$\xrightarrow{\texttt{Jan}} (\lambda x.(\lambda y.\texttt{loves}(x,y)))\ \texttt{Ann}\ \texttt{Jan}$$

Afterwards and separately, the application rule of the λ-calculus yields the desired expression to be evaluated with respect to a model, namely:

$$\texttt{loves}(\texttt{Ann},\texttt{Jan})$$

Now wait a minute, we also want semantics to be compositional, guided by the structure of syntax. But all of that structure is in the CFG. It's well understood how to do semantics for generative grammar on that end [CITE], but now we're left with a technical problem: how do we know whether the pushdown construction gives us the same semantics as the one we would usually want for the CFG? If we had that, then we would have rolled together a parser and a producer sharing the same semantics, which would be a particular mathematical model for communication. We could perform a case analysis on this example to achieve this, but that approach won't scale, and it seems pretty thorny to generalise because the whole pushdown idea operates on λ-expressions in linear form, which forgets all of the nice syntactic structure given by the CFG. If we wanted to do this systematically while respecting compositionality of semantics, it seems that the conceptually cleanest way go about it is to look for a kind of strong equivalence: we can try to make it so that every generator on the productive side corresponds precisely to an appropriate bit of partially composed semantics on the parsing side. But in this example, what are we to do with the ansatz types `S` and `VP` that live in the CFG but don't occur on the parsing side? The fundamental conceptual obstacle here is that deep structure (should it exist) potentially has access to all kinds of extra labels and structural information that the parsing side doesn't have access to; the latter can only see a string of terminals. Let's take a step back here and start over.

Maybe starting with a finite state automaton on the parsing side is not the best beginning, because it only cares about whether a string is acceptable or not. If instead we used a typelogical grammar that witnesses judgements of grammaticality along with proofs as witnesses, we could exploit the fact that the proofs themselves reflect grammatical structure. Now Section **??** is well-motivated with respect to the literature, so you can go see how this line of investigation plays out over there.

$0\frac{1}{2}$.2.5 *On frameworks for rewriting systems*

I think that Section **??** is pretty much well-motivated with respect to the literature already, so there's just one conceptual issue I want to address here. One of my examiners noted that it seems like bad mathematical taste to use free infinity-categories for what basically amounts to graph rewrites, which raises the obvious question: why not just use graph rewrites instead? First, I agree with his comment about taste, but I felt in this case it could not be avoided, because second, the answer lies in Figure **??**, and it has to do with locality and long-range dependencies. I'll try to elaborate the issue in prose here. Tree-structures represent context-free symbol-rewrite systems, and transformational grammar and tree-adjoining grammars are about rewriting trees. Of course, symbol-rewrite systems can be context-sensitive, and in the same vein, we might expect that tree- or graph-rewrite systems can also be context-sensitive. Now here is where things stop being pedestrian: suppose we have a long-range dependency of some kind between `A` and `B`, where it doesn't really matter what happens in between. An example is coreference in text. This is a kind of context-*insensitivity*, where rather than a rewrite requiring a context of a particular form, the surrounding and intervening context may freely vary to a degree. Designing an abstract machine that obeys these constraints is tricky because rewrites could in principle require evaluating whether some predicate holds for the contents of memory before executing a rewrite, as opposed to a simple lookup. Concretely, in the case of strings, context-insensitivity already requires more than just a stack-memory; it would require a Turing machine that could also operate on that memory to determine whether the context is of the right form. There's a whole host of issues here: technically it's not clear what such a machine would look like if we go from strings to graphs, and conceptually if we had such a machine we would have to craft custom limitations on the kinds of well-formedness predicates that permit rewrites so that we could stay in an efficiently parseable class of languages, on pain of our machinery overshooting the human capacity for natural language. The solution to these problems that I have opted for here is to keep all rewrites local and properly context-free for graphs, but I let the machinery of infinity categories handle bookkeeping for handling spatial rearrangements of the deep structure so that it is possible to redefine what locality means. The price to pay here is that one must explicitly keep track of not just the connectivity of the graph at any given time but also its spatial arrangement, but this is also potentially a feature: the difficulty of finding the right arrangement to go to next conceptually maps onto the computational difficulty of understanding natural language (but I won't commit to any sort of mathematical realism about it). I'll also remark that the machinery of Section **??** can be reversed to go from circuits to text, which can be piped together with the machinery of Section **??** to go from text to circuits to also give a satisfactory model of communication, though what I have not done in this case is demonstrate strong equivalence, which I will leave as an open question for now.

$0\frac{1}{2}$.2.6 *On formality in cognitive semantics*

There are two conceptual contributions here to formal linguistics. First is the demonstration that, in principle, we may interpret the meanings of words to be instructions about how to set up and move shapes around in cartoon sketches of conceptual schema, without sacrificing rigour. I do this by defining the symmetric monoidal category **ContRel** of topologically continuous relations, which serves as a mathematically formal and compositional setting for cartoons. I've gone to some length to convince the reader that, modulo a little elbow-grease, anything one can linguistically describe happening to assemblages of shapes can be modelled in **ContRel**, so we may conservatively generalise Montague semantics — construed as a symmetric monoidal functor from text circuits as an underlying deep structure to some other symmetric monoidal category as semantics — from truth-conditions to shapes.

Second is a computable (by hand) mathematical model of metaphor. The core idea is simple: if we consider a metaphor to be a pair of structure-preserving maps from the data of a conceptual schema C to two different semantic settings X and Y, so long as the maps satisfy certain algebraic properties, we may start from an expression in X, go backwards along one of the maps to C, and then travel forwards along the other map to Y to obtain another expression. The nature of the algebraic properties makes it so that travelling backwards along a map is not only well-defined, but the resolution of where we end up in the preimage also doubles as bookkeeping for different choices about how to interpret the metaphor. By starting with X as text circuits as a model of deep structure, encoding the data of a conceptual schema C as freely generated by the signature of a text circuit with equational relations, and taking Y to be the setting **ContRel**, we may compute the meaning of `Vincent spends his morning writing`, via the metaphor `TIME is MONEY`, as a vignette of pictures that represents the conceptual situation in much the same way that moving meeples around different zones in a boardgame encodes the state of the game. This sort of boardgame-semantics is then a model that serves as a further basis upon which one can do truth-conditional semantics. What is quite cute about all of this is that it is the interpretation of metaphors as pictures, using picture-maths.

These two contributions address — to my knowledge — several conceptual lacunae in both truth-conditional and vectorial semantics. The first such gap is that we use language in the first place to construct the underlying models with respect to which we might eventually apply the truth-conditional view, for instance when we tell a story, or when we present a reasoning problem in mathematics. This appears to require a discourse-theoretic approach where composition of the underlying models that truth-conditions are evaluated against, and the machinery in this thesis appears to suffice in principle. I don't doubt that there are frameworks that meld together truth-conditions discourse-theoretically; the conceptual distinction of this approach is that

I will confess that what truly motivated all of the effort in this thesis was the attempt to put metaphors on formal footing, in pictures. There is an attitude that the real mensch logic and graphs that formal linguists are fond of is somehow better by virtue of mathematical rigour than the fluffy stuff cognitive semanticists are interested in, and that didn't sit right with me. I think I've achieved what I set out to do, but in such a way that my zeal for formal linguistics annihilated itself: I was moving little blobs and stickmen around with words in a compositional fashion to create cartoon vignettes, which was really no different than programmatic animation with pen and paper under doctrinal constraints that no longer in my mind usefully restricted the nature of the kind of inquiry into language that was intended. What started as mathematical modelling in search of something essential became in myself a dawning awe of the vastness of language, and the recognition that I had both the logician's and the algebraist's sicknesses in my spirit. How I would attempt to describe them is that the logician's sickness is the desire to subjugate and quarantine the terror of reality within logos, and the algebraist's sickness is the desire to construct abstract machines that will solve the problem in a way that absolves one of having to make choices. These are, I believe, the fear of life and of living, respectively. If I went back in time, I don't think I would have been able to talk myself out of it. From discussions with other postrationalists, it seems that the sickness doubles as the sole cure: one just has to really take rationalism seriously and come out the other side with the sensibility of an artist, burdened and blessed with the responsibility to choose what is meaningful for themselves. Before, from pride, I looked upon mathematical linguistics with disdain; now, from humility, pity.

the composition of the underlying models (as states), meanings of words in text (as updates), and of truth-conditions (as tests) all all on equal footing, and all moreover in a fashion that generalises beyond truth-conditions to admit broader conceptions of what the essential data of semantics might be, as the modeller wishes.

A second gap, particularly a challenge for the vectorial view but more generally pressing, is a compositional semantics for topological linguistic relationships, such as `containment`, which motivated and justified introducing the machinery of topologically continuous relations. Evaluating/interpreting the meanings of such topological words required concrete models to evaluate against, which necessitated a construction of a compositional setting in which one could hold onto and label "non-pathological" shapes, and a bit of mathematical effort was expended to show that such collections of shapes corresponded to conservative generalisations of special frobenius algebras, which end up behaving graphically simply as wires that permit splitting and merging. That effort in my view distinguishes this approach from others in that it also sought to demonstrate an ideological point: that pictures are all you need.

The third, which is a gap for any compositional account of natural language semantics guided by syntax, is that one needs something else around in order to make sense of metaphors. For this, I argue that spans of functors — of essentially the same kind used in my analysis of communication — appear to be a promising foothold, and I do believe I am the only person thus far to have computed a pictorial semantics of a metaphor by formal means "all the way", starting from the basic syntactic structure. There are many things I am unhappy about with this thesis, but I am at least happy about that.