Distilling Grammar into Circuits (or, de-humanising grammar for efficient machine use)

Bob Coecke[†] and Vincent Wang[‡]
[†]Cambridge Quantum Computing Ltd.
[‡]Oxford University, Department of Computer Science

May 11, 2021

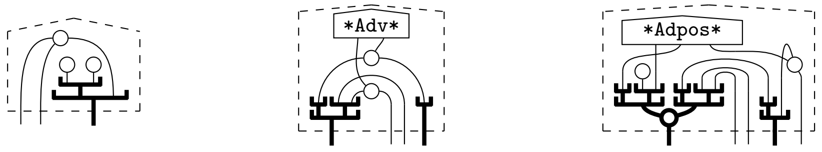
Abstract

By *dimension* we refer to the difference between one-dimensional symbolic strings and two- or higher-dimensional diagrammatic formalisms, e.g. circuits.

Much of the complexity of grammar is due to the fact that human language is a one-dimensional vehicle for higher-dimensional content. This dimensional collapse requires bureaucratic conventions and stylistic features which typically vary across different languages, such as SVO-order.

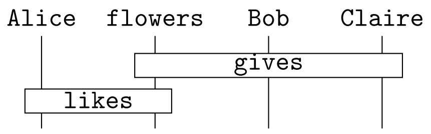
We identify *language circuits* as a lean structure for the 'factual essence of language', that is, the inner-workings of meanings within language across several layers of expressiveness—cf. words, sentences, larger text *etc.* Language circuits may capture that what is truly universal beneath grammar.

Concretely, for the specific case of pregroups, we exhibit an algorithm that transforms proofs of grammatical correctness for a substantial fragment of English into language circuits, which compose just like sentences in larger text. The crux of the algorithm consists in providing words with 'internal wirings', e.g. for a subject relative pronoun, a predicative adverb and an adposition respectively:



Together, these internal wirings form a visually depicted algorithm that undoes pregroup bureaucracy, and their complexity represents the complexity invoked by the dimensional collapse of grammar.

These internal wirings also allow us to canonically deform pregroup diagrams into language circuits, like this one:



While us humans could not verbally communicate in terms of language circuits, machines can, so language circuits can be conceived as grammar for machines.

Contents

1 Introduction

One branch of the study of the structure of language goes back deep in the previous century with the works of Ajdukiewicz [2] and Bar-Hillel [7] resulting in the Lambek calculus [36]. We call these structures 'grammatical calculi'[1]. Mathematically, what we are talking about are strings of grammatical types (a.k.a. free monoids), with some additional structure that allows one to figure out which of these strings grammatically make sense. Lambek himself went on to challenge Lambek calculus, favouring his 1999

[1]Sometimes they are also referred to as categorial grammars. While we say categorial (no C between i and a), these grammars typically also have a natural category-theoretical incarnation – see e.g. [16].

'pregroup grammars' [37, 38]. These are the ones we will also focus on in this paper. However, the core ideas of this paper extend well beyond pregroup grammars, including CCGs [55], drawing on the recent work in [63] that casts CCGs as augmented pregroups.

It has been claimed that grammatical calculus exposes the 'universality' of grammar, up to the point that some suggest grammar is hardwired in our brains. Notable names here include Chomsky [13], Montague [45], and much earlier Bacon [4]. Still, it suffices to look at different languages to see that there are very clear differences for something 'universal'. For example, for each permutation of subject, verb and object order, there is some language in the world where that 'convention' is taken.

Grammatical calculi aren't just purely academic gadgets, but have practical uses in natural language processing (NLP). For example, they allow for highly efficient parsing [15], that is, assigning grammatical roles to a string of words, and they can guide transformations of text corpora between languages, enabling general computational linguistics for languages with few data resources [9]. They were also used for combining grammar and word meanings into one compositional whole [24], resulting in a framework, called DisCoCat, that enables one to assign a meaning to a sentence given the meanings of its words. This framework is also supported by experimental implementations [28, 32], which are still ongoing [60, 44, 61].

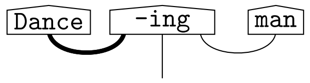
We argue that much of the complexity of grammar is due to the fact that human language is a one-dimensional vehicle for higher-dimensional content, and that this dimensional collapse comes with the introduction of:

(a) many language-dependent bureaucratic conventions, and

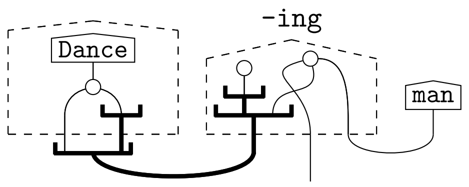
(b) introduces stylistic language features.

In this paper we put forward a procedure that distils sentences into what we call 'language circuits'. We do this for the specific case of pregroups, but our recipe can be extended to more general grammatical calculi.

We in part build further on earlier work within DisCoCat that provided so-called 'internal wirings' [51, 52, 28, 33, 21, 17, 22] for words such as adjectives, relative pronouns, and transitive verbs, and extend this to a much broader fragment of English. Diagrammatically speaking, while grammatical calculi provide wires between words in order to elucidate their interactions, we also provide wirings within words. For example, a pregroup diagram for the phrase:



will become:



These additional wirings will generate an equivalence relation on sentences that equates them up to features/bugs (a) and (b) mentioned above.

We introduced these *grammar equations* in the earlier conference paper [25]. Previously additional structure for grammatical calculi had been introduced by providing semantics to certain words, for example, quantifiers within Montague semantics [46]. This is not what we do. Grammar equations strictly stay within the realm of grammar, and grammar only, and hence provide a genuine refinement of the theory of grammar, introducing novel 'grammatical truths'. Here we recall that story, and expand upon it.

These internal wirings will allow us to canonically deform the pregroup representations of grammar into circuits, as the circuits provide a *normal form* for the grammar equations. An important technical feature of our work is that the passage to language circuits cannot be done in terms of grammatical calculi in the form of preordered monoids, but requires diagrams in a fundamental manner, and in fact, as also shown in [25], internal wirings also make no sense in term of preordered monoids. Hence passing to the realm of diagrammatic representations – which correspond to proper free monoidal categories – is not just a convenience, but a necessity for this work.

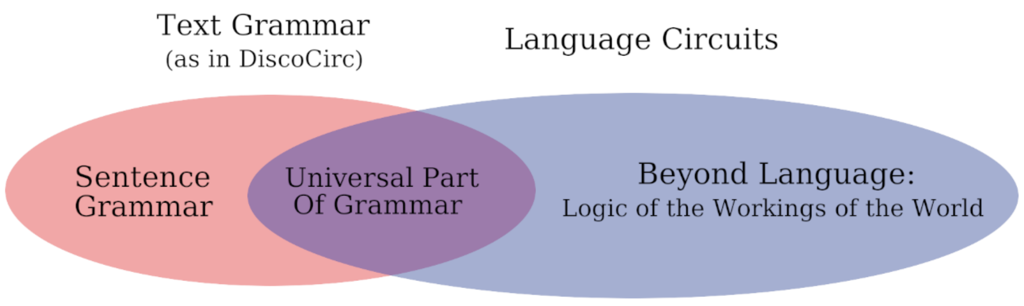
A simplified account of grammar will obviously make any task were grammar comes into play easier, and in particular when machines are involved. It can bring grammar into play in a manner it isn't yet, for example, within machine learning driven natural language processing. However, the result of canonically merging vector embeddings for meanings with grammatical calculi results in a formalism that closely resembles quantum theory [14, 19]. This means that straightforward implementation will result in a potential exponential blowup of required space resources.

Addressing this hurdle, more recently DisCoCat was successfully implemented on quantum hardware [43, 41], under the name QNLP. This was the first time that NLP was done on quantum hardware, and it was done using full sentences in a grammar-aware fashion, using corpora consisting of over 100 sentences. Given the size and limitations of the currently available quantum hardware, it is surprising that anything like that was even possible. In our hardware implementations of QNLP we were already implicitly using the ideas of this paper, and we attribute the success of that endeavour to doing so.

A more recent variant of DisCoCat called DisCoCirc extended the DisCoCat framework to larger text consisting of multiple sentences [17]. One particular feature of DisCoCat has always been an elegant diagrammatic presentation, using a formalism borrowed from category theory [23, 20], and in the case of DisCoCirc text takes the form of circuits. Implicitly and informally, it adopted a simplified representation of grammar like the one we introduce here, and this intuition will be the starting point for this paper.

We also believe that language circuits are not just a representation of structures in language, but that this structure in language is itself a representation of the workings of the world 'out there'. More precisely, we believe that language circuits for an essential part of the interface that relates our thoughts to the outside world, or using AI terminology, it is an essential part of our embodiment. Hence we expect it to play a role in

AI well beyond NLP. Here's how for us things relate:



To conclude, before even starting, with language circuits we are bringing forward an essential structure of reality, which is present in language, and in fact, possibly at the origin of the universal structure of language. In case you feel this is too strongly stated, we do also put forward an essential structure of language.

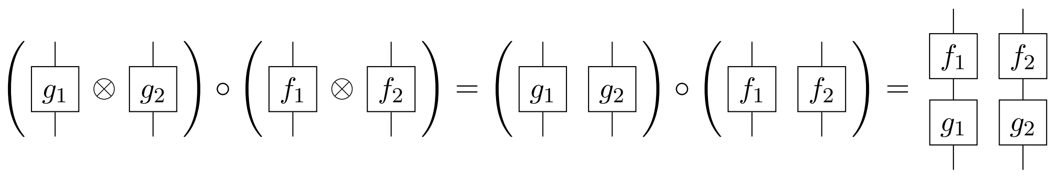
We discuss related work in Section 8. This includes previous work within DisCoCat [28, 34, 51, 52, 33, 21, 17, 48, 59, 26, 40], relations to discourse representation structures [47, 31, 18, 57], Harris and Chomsky's transformational grammar [50], Chomsky's deep structure [8], dynamic semantics [30, 58, 53], dynamic epistemic logic [6, 5], other logic-oriented approaches to text [3, 42], and dependency grammars [56].

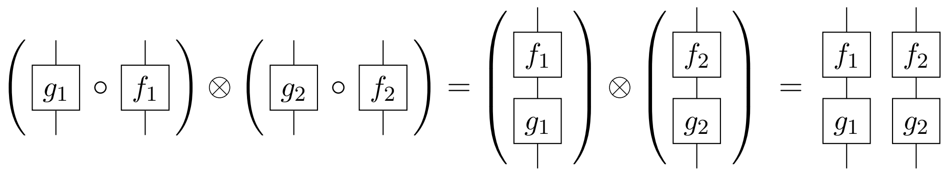
2 The natural habitat for language

Sometimes things are forced to live in a world that is not their natural habitat. One such thing, in mathematics, are tensor categories, also known as tensor networks. This structure plays a central role in fields like quantum theory [20], computer science [1], and many others, capturing the operations of parallel composition $\otimes$ (*a.k.a.* 'while') and sequential composition $\circ$ (*a.k.a.* 'after'). Typically, in textbooks, this structure is represented by one-dimensional (1D) strings of algebraic symbols, and then, equations are needed to express their interaction, most notably, 'bifunctoriality':

$$(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2)$$

On the other hand, representing it in terms of 2D diagrams, reading diagrams from top to bottom (as we do in English), no equations are needed:





That is, by using the 2D format of diagrams, the defining symbolic equations are built-into the geometry of the plane. The reason for the symbolic representation being more involved is that in order to force something two-dimensional (2D) on a 1D line, artificial bureaucracy needs to be introduced (*e.g.* bracketing) and this requires extra rules as well. A more detailed discussion of all of this can be found in [23, 20].

Something similar, although a bit more subtle, is true for language. In order to see this, it suffices to look across different languages, where different choices have been made about word-ordering. For example, all possible permutations of subject-object-verb (SOV) order occur in some existing language, *e.g.* in French and English:

`je t'aime` (SOV) vs. `I love you` (SVO)

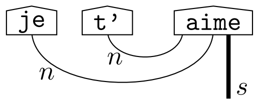
Given that these sentences have the same meaning, the word-ordering is really not much more than meaningless bureaucracy. In other words, the natural habitat for the words in these sentences is not a 1D ordered triple. We emphasise 1D here, as human communication by means of speech is intrinsically 1D.

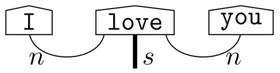
Grammatical calculi [36, 29, 37], which are a mathematical account of grammar, reflect that 1D-ness. Their core is a string-generating 'monoid', for example, in symbolic 'pregroup' terms [38] the grammar of the above two sentences respectively is:

$$\underbrace{n}_{S} \cdot \underbrace{n}_{O} \cdot \underbrace{\left({}^{-1}n \cdot {}^{-1}n \cdot s\right)}_{V} \qquad \text{vs.} \qquad \underbrace{n}_{S} \cdot \underbrace{\left({}^{-1}n \cdot s \cdot n^{-1}\right)}_{V} \cdot \underbrace{n}_{O}$$

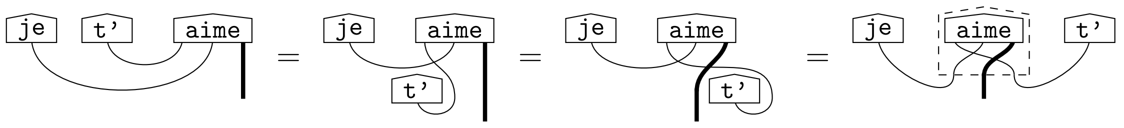
Where n stands for noun and s stands for sentence.

In corresponding diagrammatic terms [49, 24] (which we explain in more detail in Section 4, and which in this paper is vital) we see very different nesting patterns when looking at how the subject and the object meanings 'enter' the transitive verb:[2]





On the other hand, the connective structure is the same, and we can <u>deform</u> one diagram into the other, provided we allow words to leave the 1D line:

 (1)

This example tells us that the natural habitat for language isn't 1D.

This phenomenon also happens within a single language, with different word-orderings carrying the same factual content. Consider the following example:

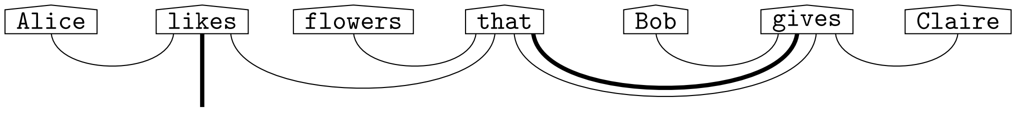
`Alice likes the flowers that Bob gives Claire`

vs.

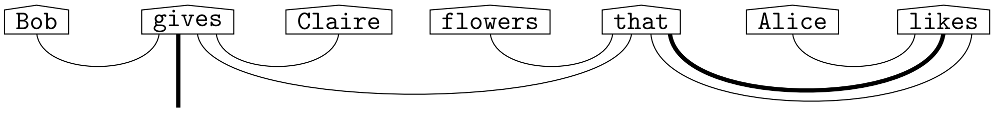
`Bob gives Claire the flowers that Alice likes`

[2]As we explain in Section 4, the diagrams do not only represent the grammatical types, but also the derivation using pregroup algebra that the sentence is well-formed.

The pregroup diagrams now look as follows:



vs.



The factual content which these two sentences convey, in absence of any additional context, is exactly the same. The difference that may occur within specific contexts is that the position of the verbs `likes` and `gives` with respect to the relative pronoun `that` comes with a built-in *potential* to indicate a causal dependency of the first part of the sentence on the second part. For instance, `Alice` may `like the flowers that Bob gives Claire` *because* `Alice` and `Claire` are friends, and `Bob` may have `given the Claire flowers that Alices likes` *b*ecause `Bob` wants to spite `Alice`.

Is there any way that enables us to derive the close correspondence between those two sentences? Moreover, is there any way to relate them that also accounts for the difference in terms of causality that may occur when they are placed within a larger context?

The algorithm we put forward in this paper will enable us to relate them, while also capturing the potential causal differences that may exist between them. However, the deformation trick (1) wouldn't do the job this time. More needs to be done in order to identify these sentences as effectively the same. We will do so in Section 4.

Altogether, due to the 1D restriction, there seems to be a lot bureaucracy at play within grammar. So we ask ourselves:

$$\boxed{\textit{What is the natural habitat of language?}}$$

Along the same lines, while it has been claimed that grammar is supposed to be universal across all languages [13, 45, 4], grammatical calculi don't reflect that universality given that different languages have different grammars. Hence:

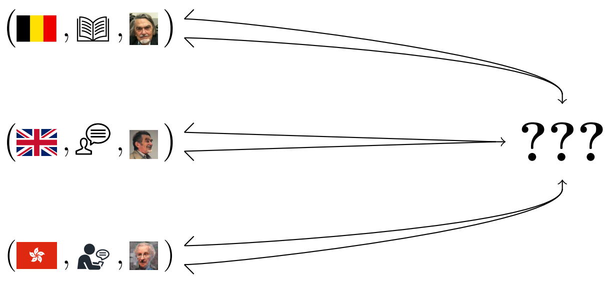
$$\boxed{\textit{What is truly universal about grammar?}}$$

Besides the academic interest of the above questions, they are also also practically important as we don't want to impose a burden on machines that is really just a consequence of our own oral and verbal limitations.

We will propose an algorithm called 'distillation', that addresses these questions:

$$\text{gram. calc.} \xrightarrow{\text{distillation}} \textbf{???}$$

and in particular, propose what **???** should be. Drawing from our discussion above, a grammatical calculus formally captures grammar, and in doing so reflects language-dependent conventions (icon ▮), stylistic features (icon 📖), and specific features of the particular grammar calculus used to formally represent the grammar (icon). Distillation means to extract the factual content, eliminating these differences:



and we want the product of distillation to be that which is *essential* to language.

3 Going 2D: language circuits informally

As mentioned in the introduction, in this paper we focus on pregroup grammatical calculi (for English), not only because of their elegant diagrammatic presentation that will allow the conversion into language circuits to be purely topological, but also since they are the simplest of all grammatical calculi, and can be refined into other ones including CCGs [55], drawing on the recent work in [63].

As mentioned in the introduction, DisCoCat [24] was introduced in order to compute the meaning of a whole, from the meaning of its parts, given grammatical structure. Concretely, DisCoCat decorated grammar with meanings, via a functor:

$$\mathsf{DisCoCat} : \mathsf{Pregroup\ Diagrams} \longrightarrow \mathsf{Meaning}$$

DisCoCirc [17] was introduced in order to extend DisCoCat to text, and in particular, in DisCoCirc sentences can be composed, resulting in a 2D circuit.[3] So having textual composition seems to take us out of a 1D realm. Something else DisCoCirc does is to update meanings as text progresses, repeating a slogan from [17]:

$$\boxed{\textit{A sentence is a process that updates meanings}}$$

We hypothesise here that the structure of these updates really matches what we were looking for in the previous section, namely that what is truly universal about grammar.

Let's consider again the sentence:

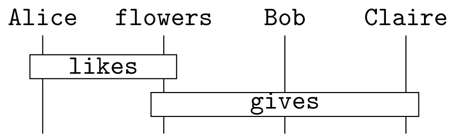
`Alice likes the flowers that Bob gave Claire`

and see how the meanings of words are altered by it. Factually, we learned two things:

- Concerning Alice and the flowers, that there's a `liking`-relationship.
- Concerning Bob, the flowers and Claire, that there is a `giving`-relationship.

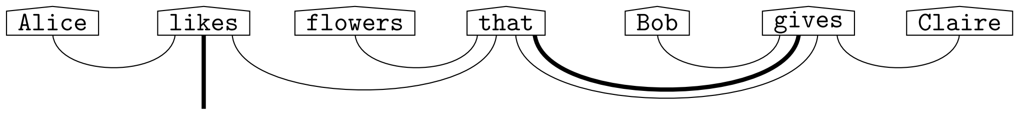
[3]The 1D vs. 2D angle may sound a bit confusing at first, since DisCoCat diagrams are also 2D. Still, in such a diagram the words are all restricted to a line, placed side-by-side.

We can now represent this as a circuit, with the 'actors' involved as the wires of the circuit, and the 'meaning-updating processes' as gates:



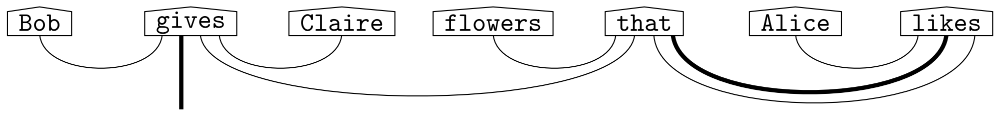
(2)

Now compare this circuit with the sentence's pregroup diagram:

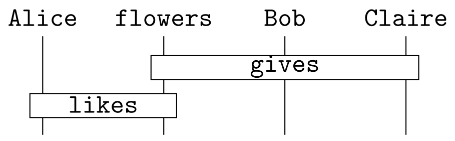


A remarkable simplification occurred: the circuit is visually natural, much simpler to replicate from memory than the pregroup diagram. This hints at the fact that something (cf , and from the previous section) has been removed.

Now let us consider the following clearly different pregroup diagram:



We obtain a very similar circuit:



(3)

The difference between (2) and (3) is the ordering of the gates. This can be interpreted in different manners, all compatible with our approach:

(α) If there is no causal relationship between the two parts of the sentence, then we expect the gates to commute, so the two circuits are equal.

(β) If there is causal relationship between the two parts of the sentence, the difference in the ordering of the gates is a witness of that fact.

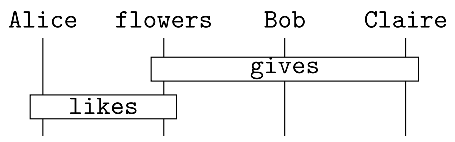
Other options concerning the circuit could include:

(γ) Not treating all actors on the same footing, for example, one may want to treat `flowers` differently than `Alice`, `Bob` and `Claire`. This would be possible, following [17]. For clarity of the argument in this paper we treat all 'actors' on equal footing.

Now consider the text:

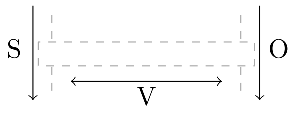
`Bob gives Claire flowers`
`Alice likes those flowers`

Again we get this circuit:



However, now it is a circuit representing two sentences, so we are now dealing with a grammar that composes 'text', which is of course not at all surprising given that we took DisCoCirc as our starting point. Yet, it is a clear departure from traditional grammatical calculus, and only enabled thanks to the passage to 2D circuits [17].

Let's now have a closer look at how 2D gets exploited in order to simplify things as compared to pregroup diagrams. For a single gate with two inputs, which we can think of as a transitive verb, the very fact that the gate has two inputs already tells us that a subject and an object are needed. In particular, these inputs enter at the top of the gate, and leave at the bottom, which is orthogonal to the direction along which the transitive verb relates the subject and the object:



so we are indeed exploiting 2D.[4]

4 From pregroups to language circuits

For our purposes, a pregroup has a set of 'basic types' $n, s, ...$ each of which admit left and right inverses ${}^{-1}n$ and n^{-1}. Each grammatical type is assigned a string of these, e.g. a noun gets n, and a transitive verb (in English) gets:

$$tv = {}^{-1}n \cdot s \cdot n^{-1}$$

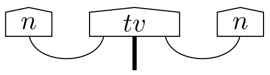
The inverses 'cancel out' from one direction:

$$n \cdot {}^{-1}n \ \rightarrow\ 1 \qquad\qquad n^{-1} \cdot n \ \rightarrow\ 1 \tag{4}$$

A sentence is grammatical if when taking the string of all of its grammatical types, the inverses cancel to leave a special, 'final', basic type s (for sentence), like here for $n \cdot tv \cdot n$:

$$n \cdot \left({}^{-1}n \cdot s \cdot n^{-1}\right) \cdot n \ \overset{(assoc.)}{\rightarrow}\ \left(n \cdot {}^{-1}n\right) \cdot s \cdot \left(n^{-1} \cdot n\right) \ \overset{(4)}{\rightarrow}\ 1 \cdot s \cdot 1 \ \overset{(unit)}{\rightarrow}\ s$$

This calculation can be represented diagrammatically:



(5)

where:

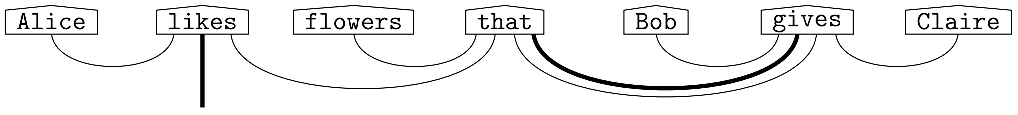
[4]Although, as specialists will tell you, rather even 4D, given that we allow wires to freely move past each other, such that we only care about their *topological connectedness*.

- the boxes are the types we start from,
- the 'cups' represent the cancellations (4),
- the straight wire corresponds to types that aren't cancelled, and
- the associativity step (assoc.) and elimination of the unit step (unit) become trivial when using diagrams, just like it was the case for bifunctoriality in Section 2.

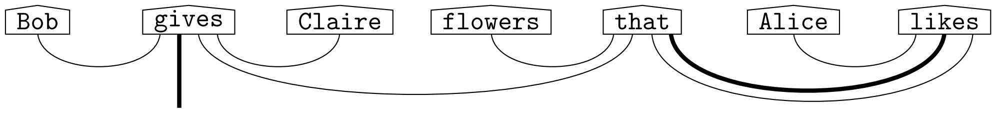
Definition 4.1. A calculation like (5) is called a pregroup proof.

In (5) we want to think of the two nouns being fed into the transitive verb, which then puts out the meaning of the entire sentence. A more detailed discussion is in [17].

Earlier we saw these two pregroup diagrams:



(6)



(7)

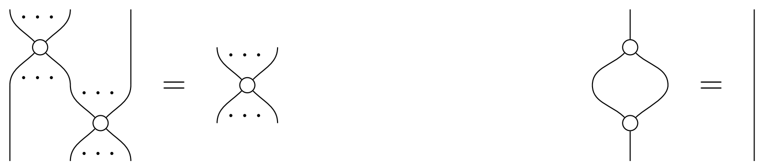
which couldn't be related to each other as things stand. We would like to know what is needed to be able to do so. We also want to know what is needed to be able to turn them into the circuits (2) and (3). As it turns out 'what is needed' coincides in both cases.

4.1 Rewriting pregroup diagrams via internal wirings

What is needed are 'internal wirings' [51, 52, 28, 33, 21, 17, 22] of certain words, that is, not treating these as 'black boxes', but specifying what is inside, at least to some extent. Equationally speaking, they provide a congruence for pregroup diagrams, and we can establish equality by means of topological deformation.

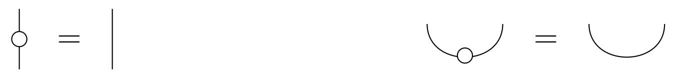
4.1.1 Spiders

For constructing these internal wirings we make use of 'spiders' [20] (a.k.a. Frobenius algebras [12, 23]). One can think of these spiders as a generalisation of wires to multi-wires, as rather than having two ends, they can have multiple ends. Still, all they do, like wires, is connect stuff, and if one connects connected stuff to other connected stuff, then everything becomes connected (a.k.a. 'spider-fusion'):

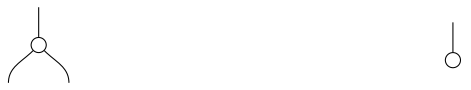


(8)

The plain wires and cups we saw above are special cases:

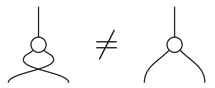


and other examples include 'copy' and 'delete':



(9)

In general, spiders can be non-commutative, that is:

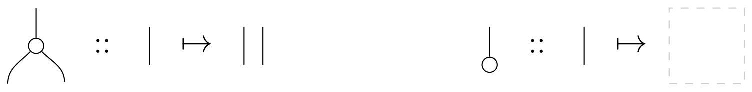


which are necessary when (cf. the discussion in Section 3):

(α) there are clear causal relationships, and/or

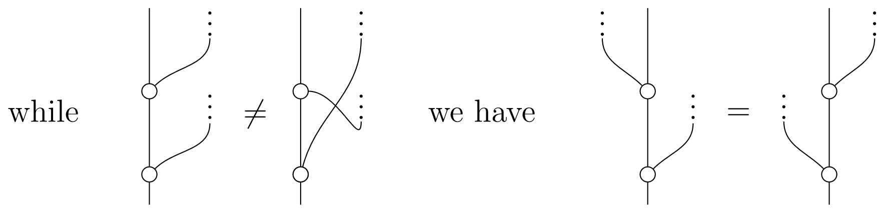
(β) one wants to account for differences in emphasis.

Remark 4.2. What's being copied by (9) are wires:



When these diagrams also carry meaning, like in DisCoCat, then the copy spider typically won't copy the meaning vectors, but rather create correlated pairs. This is akin to the quantum no-cloning theorem.

An important property for these non-commutative spiders is:[5]



(10)

One could put this as a slogan:

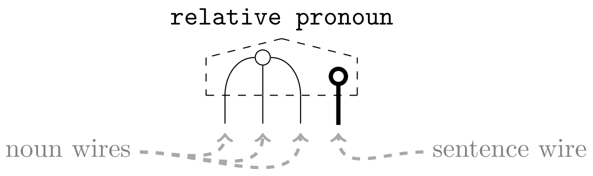
$\boxed{\textit{spider heads can move past each other, but spider legs can't.}}$

Remark 4.3. As shown in [25], internal wirings in terms of spiders don't make sense for symbolically defined pregroups (i.e. preordered monoids) since the spiders force symbolic pregroups to be trivial. A passage to certain (free) monoidal categories is needed, like in [49], but of course, as always we will work with the diagrammatic representation of these [54, 23], that is, pregroup diagrams.

[5]The first equation requires commutativity to be valid while the second is about associativity, a property that follows from (8).

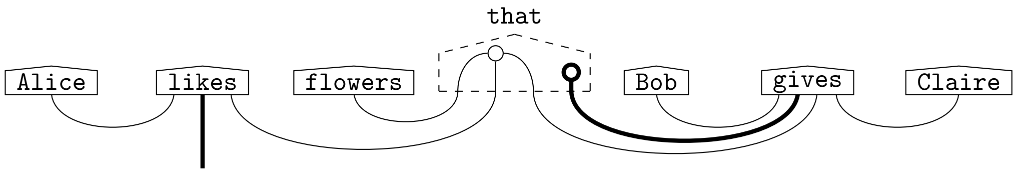
4.1.2 Internal wiring for relative pronouns

For relative pronouns we start with the internal wirings that were introduced in [51, 52]:

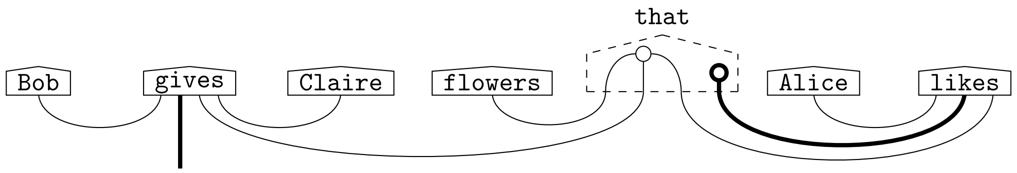


(11)

Substituting this internal wiring in the pregroup diagrams we saw above:

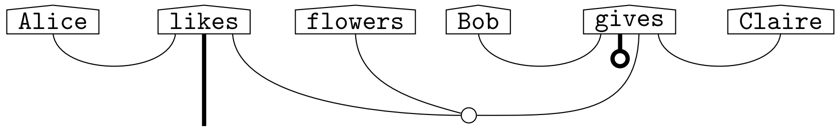


(12)

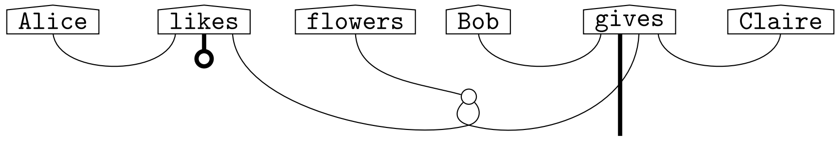


(13)

permuting the boxes a bit, more specifically, swapping `Bob gives Claire` and `Alice likes` in the 2nd diagram, the two diagrams start to look a lot more like each other:



(14)



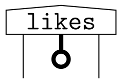
(15)

Their only difference is a twist which vanishes if we take spiders to be commutative, and either a loose sentence-type wire coming out of the verb `likes` in the first diagram, versus a loose sentence-type wire coming out the verb `give` in the second diagram, the other verb having its sentence type deleted.

Remark 4.4. Note also how we could now think of `flowers` being 'copied' by the spider and one 'copy' is then provided to each of the verbs. A better view is that a single copy is shared by the verbs — see Remark 4.2.

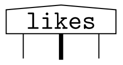
4.1.3 Internal wiring for verbs

The deleting of sentence-types of verbs:



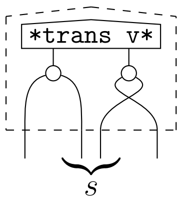
(16)

by the internal wiring of relative pronouns seems to prevent us from bringing the diagrams (14) and (15) any closer to each other, as in general there is no way to get back to:



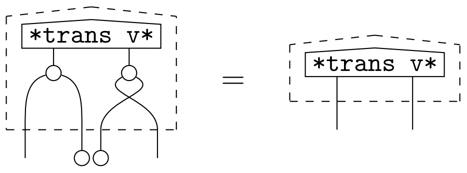
However, this irreversibility does not happen for a particular kind of internal wiring for the verb, used in a number of earlier papers [28, 33, 17, 22]. In those papers spiders were assumed to be commutative, while here we need them to be non-commutative. We arrange a twist such that the right legs of the spiders contribute to the sentence type.

Definition 4.5. A transitive verb comes in spider-form if for some choice of spiders it has internal wiring:

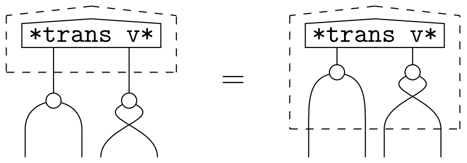


(17)

For transitive verbs in spider-form, if the sentence type gets deleted:



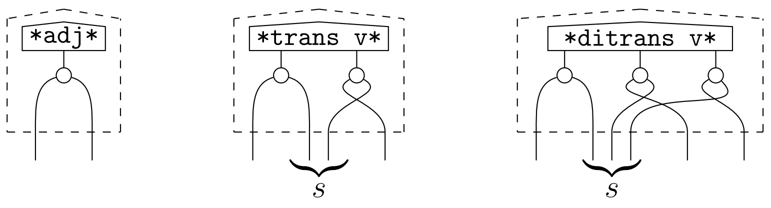
we can bring back the original form by copying the remaining wires:



So nothing was ever lost. To conclude, for the internal wiring of verbs proposed above, the copying and deleting spiders now guarantee that in (16) nothing gets lost.

Remark 4.6. This presence of the operations copying and deleting is not dissimilar from how one takes linear logic and turns it into ordinary classical logic by adjoining the ability to copy and delete premisses [27]. Similarly, here we bring in copying and deleting on top of the linear structure of pregroups in order to achieve the rewriting of pregroup diagrams that are related to each other.

The spider-form extends to adjectives, transitive verbs and ditransitive verbs:

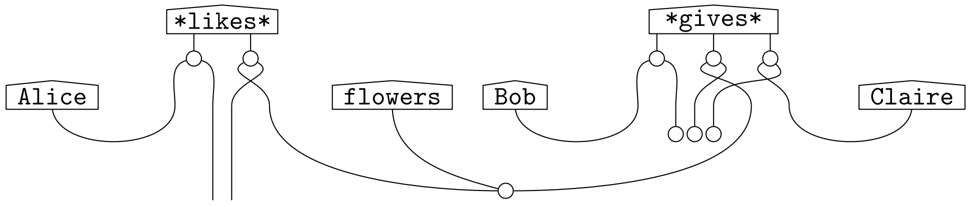


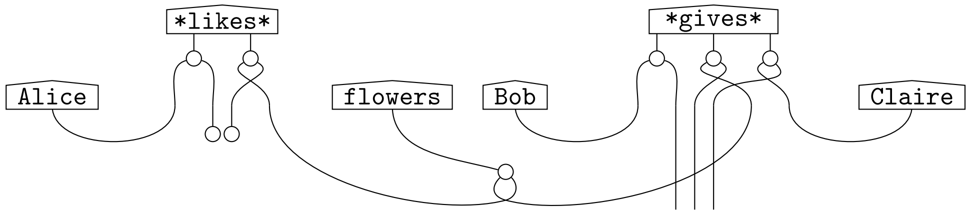
(18)

and the same principle applies here too.

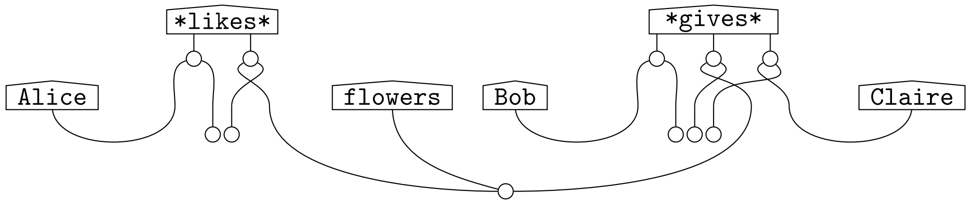
4.2 Rewriting pregroup diagrams into each other

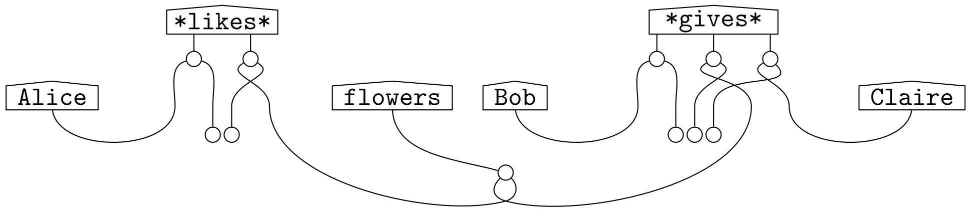
Let's look again at our example sentences, now with internal wirings (18):



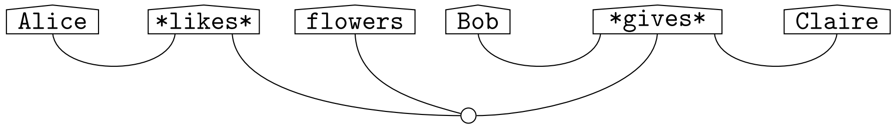


Then deleting all the outputs we get:

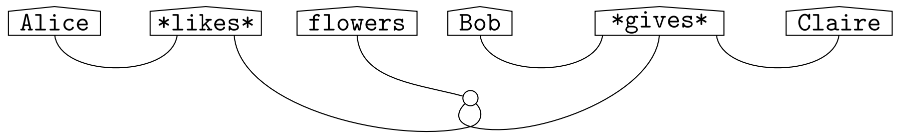




Using the spider-fusion (8) we can further simplify these:



(19)

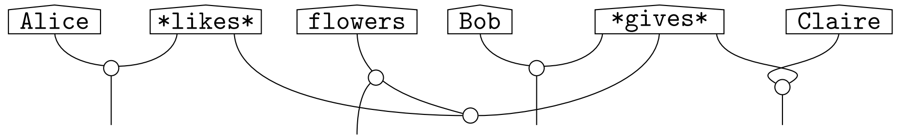


(20)

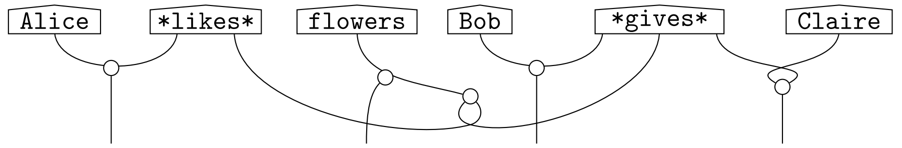
The only difference is now the twist. If there is no causal relationship between the two parts involving `likes` and `gives` (cf. (α)) the spiders will act commutatively on them, and the two pregroup diagrams are the same. If there is a causal relationship (cf. (β)) then the twist witnesses this.

4.3 Rewriting pregroup diagrams into circuits

However, we have no outputs anymore, so let's just stick in a copy-spider (to the right) for all nouns, and then we respectively obtain:

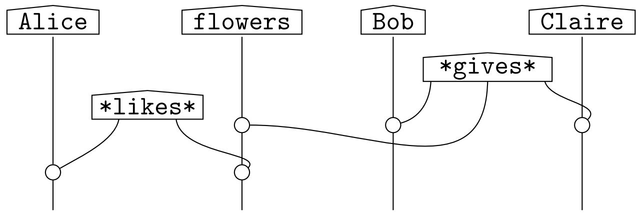


(21)

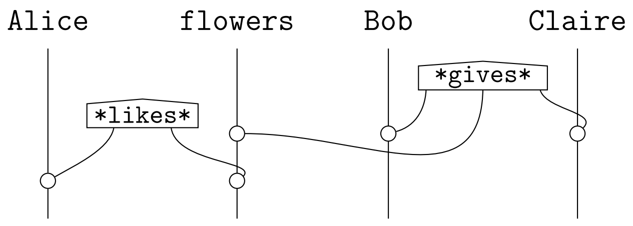


(22)

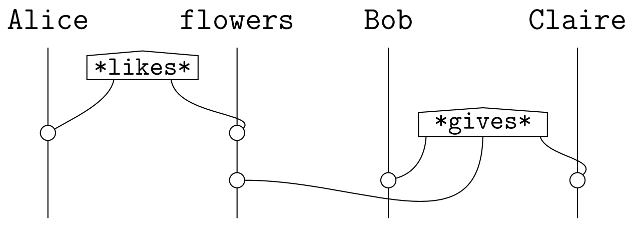
Now deforming (21) and using spider-fusion – and (10) in particular, which allows the copy-spider attached to `flowers` to slide past the other spider – gives us:



Replacing these boxes by names we get a true circuit that can be composed:



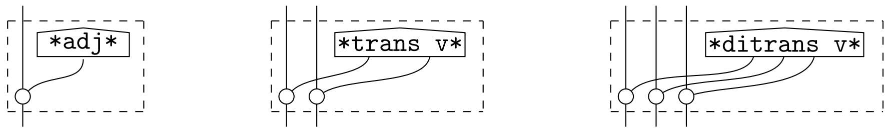
In the case of (22) we obtain:



(23)

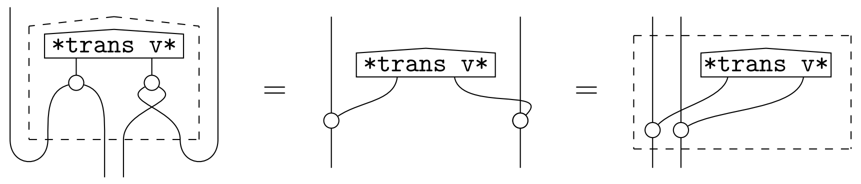
Hence, when the two diagrams representing our example sentences become closely related, they also admit a circuit form, which we call language circuits.

Comparing now to circuits (2) and (3), the order of the gates is indeed reversed, and we see that non-commutativity of spiders corresponds to non-commutativity of the gates. This is not surprising. Above the gates take the following shape [22]:



(24)

Which we can also think of as internal wirings, induced by the ones we assumed in (18):



(25)

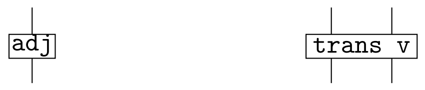
Definition 4.7. Gates in language circuits come in spider-form if for some choice of spiders they have internal wiring (24).

4.4 Innocence of spider-forms

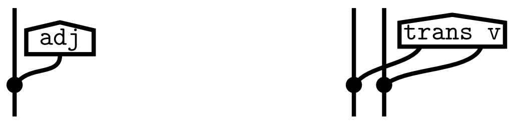
So we have left it open what the spiders actually are. There is one particularly interesting choice, which shows that we can obtain the spider-form at no cost whatsoever.

Lemma 4.8. Without loss of generality, we can assume that gates in language circuits come in spider-form (24) for an appropriate choice of wires and (non-commutative) spiders. Equivalently, we can assume that adjectives, intransitive, transitive, and ditransitive verbs respectively have the internal wirings (18).

Proof. We provide the proof in the case of gates, which then translates to adjectives, intransitive, transitive, and ditransitive verbs through (25). Given an adjective and transitive verb as a gate:



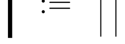
we can put them in the form:



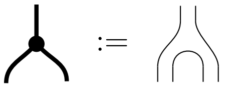
(26)

as follows:

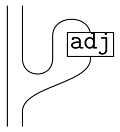
(a) one doubles the wires:



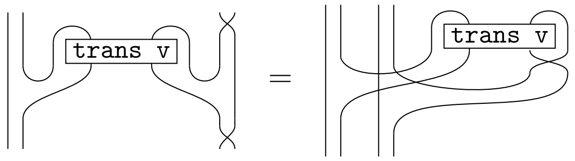
(b) one uses the (well-known) pair-of-pants spiders:



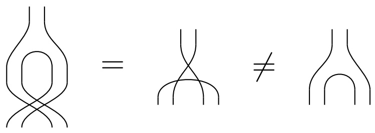
Then we can cast the adjective and transitive verb respectively as:



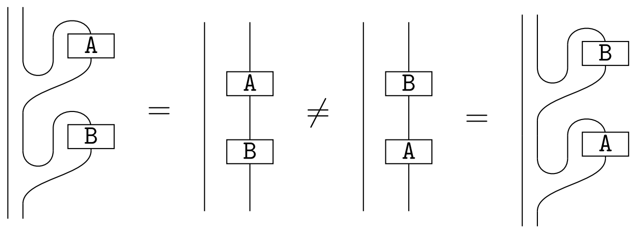
and:

 (27)

The same trick extends to intransitive and ditransitive verbs. Pair-of-pants spiders are non-commutative. Indeed, if we swap the two legs we get:

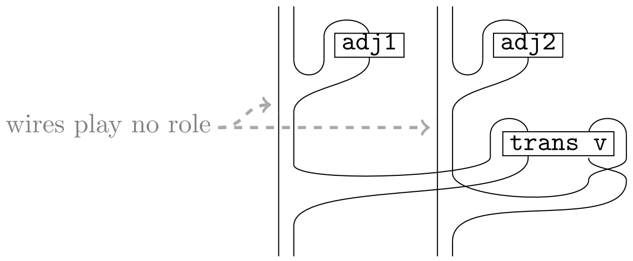


and this leads to non-commutativity of the corresponding gates e.g.:



since the boxes are typically non-commutative. □

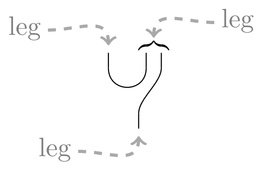
Remark 4.9. When composing gates of the form (26) in a circuit, like here:



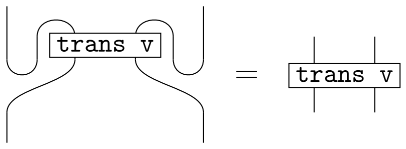
it immediately follows that the left wire is effectively doing nothing. So we may as well ignore it, and instead of (a) and (b) as in the proof of Lemma 4.8 we could:

(a) leave the wires as they are, and

(b) use the following 'pseudo-spiders':



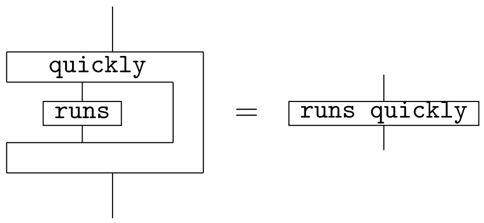
Without those 'numb' wires, for example, the gate (27) now simply looks as follows:



i.e. like regular gates.

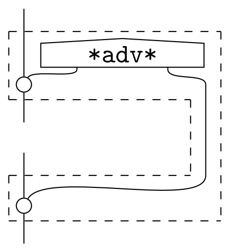
4.5 Gates acting on Gates

Thus far we have represented nouns as wires, and verbs as gates that act on these wires. An adverb like `quickly` turns a verb like `runs` into another verb, namely `runs quickly`. Hence we expect adverbs to turn a gate into another gate:



In circuit jargon these are sometimes referred to as combs. Just as in the case of gates we propose a spider-form for adverbs and other combs:

Definition 4.10. An adverbs comes in spider-form if it has internal wiring:

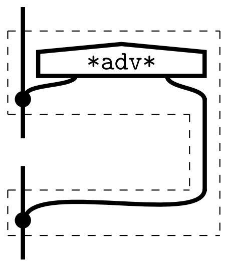


(28)

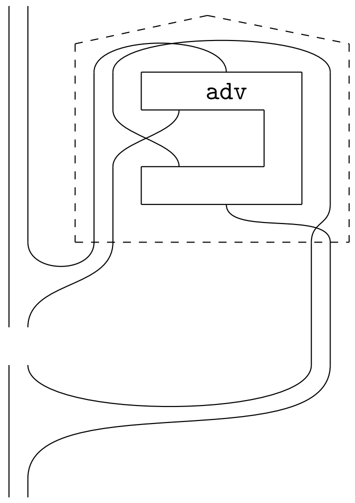
And just as in the case of gates this choice is innocent:

Lemma 4.11. Without loss of generality, we can assume that combs such as adverbs in language circuits come in spider-forms like (28) for an appropriate choice of wires and (non-commutative) spiders.

Proof. By again relying on the pair-of-pants spiders as we did in Lemma 4.8 for adjectives and verbs we can put an adverb in the form:



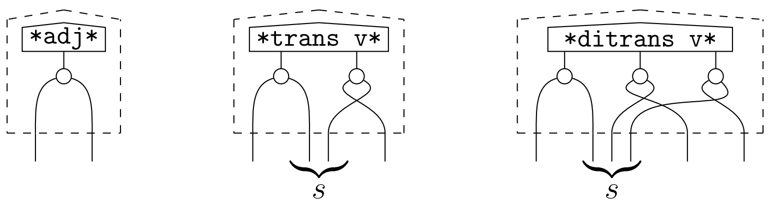
when we cast it as:



and the same trick extends to combs taking in transitive and ditransitive verbs. □

5 The wrapping gadget

Above in Section 4.1.3 we saw that sentence wires were decomposed into noun wires. However, for pregroup proofs it is important to know that those wires do belong together, which we informally indicated as follows:



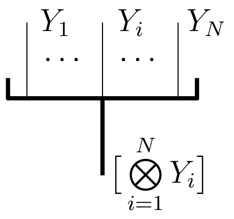
So using standard logic notation, our types are as follows.

Definition 5.1. The wire-types are generated as follows:

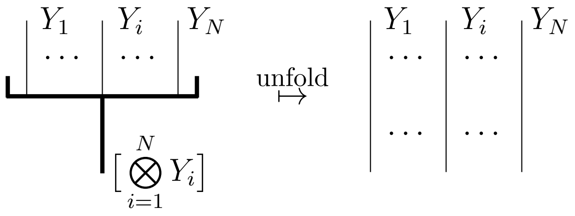
$$Y = n \mid Y \otimes Y \mid [Y]$$

Here, $Y \otimes Y$ means putting wires side-by-side, while $[Y]$ means considering wires together. In fact, Lambek [38] also used square brackets to additionally restrict the proofs of grammaticality pregroups allow for, in order to avoid ambiguities. We now provide a formal counterpart to bracketting in terms of diagrams.

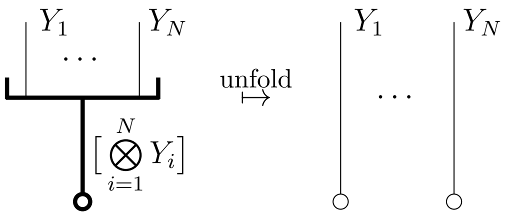
Definition 5.2. The wrapping gadget forces a number of wires to be treated as one, i.e. it wraps them, and is denoted as follows:



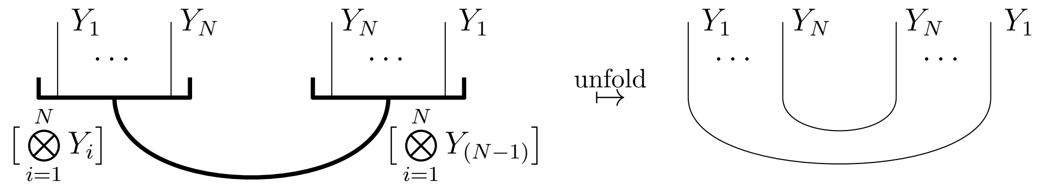
By unfolding we mean dropping the restrictions imposed by the wrapping gadget:



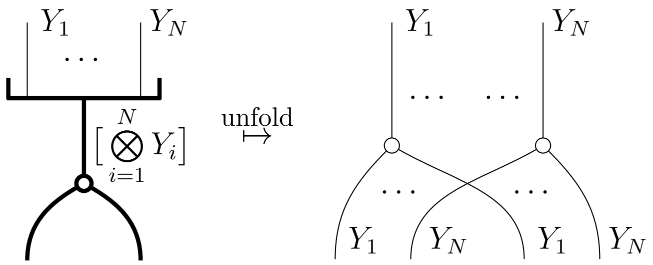
Cups and spiders carry over to wrapped wires in the obvious way, e.g.:

 (29)

with the following conventions being made for composites of cups and copy-spiders:

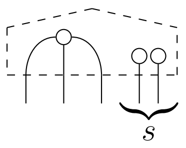


Y_1
Y_N
Y_N
Y_1
$[\bigotimes_{i=1}^{N} Y_i]$
$[\bigotimes_{i=1}^{N} Y_{(N-1)}]$
unfold

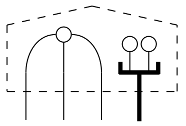


Note that when considering cups as spiders the convention for composites doesn't match the one we made. Indeed, as spiders the type would be $[Y_1 \otimes Y_N] \otimes [Y_1 \otimes Y_N]$, which would require cups to cross, versus $[Y_1 \otimes Y_N] \otimes [Y_N \otimes Y_1]$ in our convention. The reason for doing so is that our convention for cups matches what happens when doing proofs in pregroups, and also makes diagrams more readable because of the cups being nested. However, this convention cannot be extended to arbitrary spiders.[6]

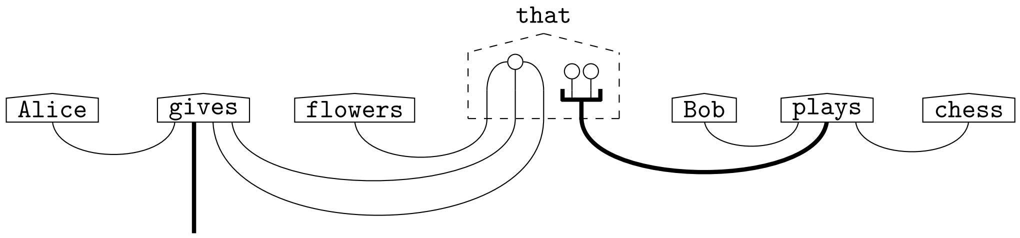
We illustrate all this now on the example of relative pronouns seen earlier. For:



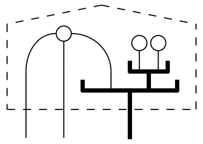
where the sentence wire consists of two noun-wires this becomes:



However, that's not enough, as we can now produce the following diagrammatically represented proof of grammaticality that abuses the object relative pronoun:



which obviously we don't want. The reason is that the deleting of the sentence type of `plays` belongs together with the noun-wire now connecting the relative pronoun with `gives`, like in (12). This is imposed as follows:


(30)

Hence, in the case of the internal wiring of relative pronouns, we do not only need one, but two wrappings. In order for pregroup diagrams to match these wrapped wires, they will also need wrapping gadgets and wrapped wires.

Definition 5.3. A pregroup diagram with wrapping is a pregroup diagram which in addition may contain wrapping gadgets and wrapped wires.

[6]The same mixed convention is also made in [20].

6 A catalog of internal wirings

We now provide a catalog of internal wirings for a substantial fragment of English. Together, these internal wirings form a visually depicted algorithm that undoes pregroup bureaucracy, and their complexity represents the complexity invoked by the dimensional collapse of grammar.

We divide these internal wirings in on the one hand 'content words', like (18) in the case of adjectives and verbs, and on the other hand 'functional words', like (11) in the case of relative pronouns, the difference being that:

- in the case of content words there will still be a 'black box', but with less wires,
- while in the case of functional words there only will be wiring.

For the cases of adjectives, verbs and relative pronouns:

- we make necessary improvements as compared to the previously established internal wirings [51, 52, 28, 33, 21, 17] by introducing wrapping gadgets just as we did in the previous section for relative pronouns – cf. (30).

We make a number of simplifications as well:

- We ignore determiners such as `the` and `a`. We could easily consider them, but that would distract from our core message by complicating diagrams.
- We ignore tenses of verbs. Tenses can be handled by introducing subtypes [38], that is, diagrammatically, having different kinds of basic wires of different types. We can also recover case agreement for plurals and gender in the same way.

A special case includes:

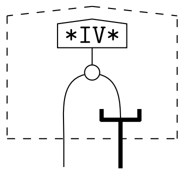
- nominative pronouns like `she`, `it`, `they` which will be treated as regular nouns in the pregroup diagrams. We describe who/what these pronouns are referring to at the level of language circuits.

We also provide the pregroup typings of each of the words.

6.1 Content words

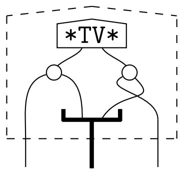
Verbs We've already seen these, but now we present then with the wrapping gadget. We stop at ditransitive verbs, as tritransitive verbs are very rare in any natural language. In principle there isn't a reason to stop at three wires, and although this may be hard for us humans to handle, as we will see, we can build verb-like boxes with more wires using just intransitive and transitive verbs plus adpositions. We can also construct ditransitive verbs in this way. The ordering of arguments for ditranstive verbs in English, from left to right, is *subject-object-theme* where object and theme follow the verb.

`Intransitive Verb`



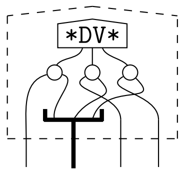
$^{-1}n \cdot [n]$

`Transitive Verb`



$^{-1}n \cdot [n \cdot n] \cdot n^{-1}$

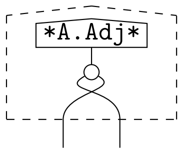
`Ditransitive Verb`



$^{-1}n \cdot [n \cdot n \cdot n] \cdot n^{-1} \cdot n^{-1}$

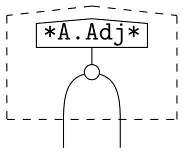
Adjectives. We consider two kinds of adjectives, depending on whether they appear before or after the noun it modifies. In the latter case in `the car is purple` we treat all of `is purple` as the adjective, ignoring the copula `is`.

`Attributive Adjective`



$n \cdot n^{-1}$

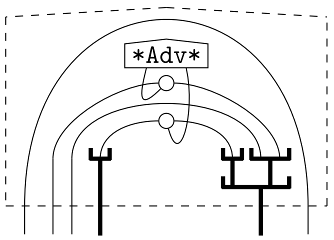
`Predicative Adjective`



$^{-1}n \cdot n$

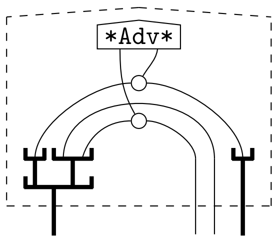
Adverbs. Adverbs admit the same attributive/predicative distinction as adjectives, depending on whether they modify verbs before or after cf. `quickly runs` vs. `runs quickly`, and we also have to consider the different arities of the verb.

`Attributive Adverb`$_{\texttt{IV}}$



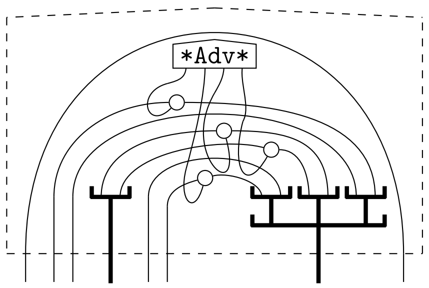
$^{-1}n \cdot n \cdot {}^{-1}n \cdot [n] \cdot [[n \cdot {}^{-1}n] \cdot [n]]^{-1} \cdot n$

`Predicative Adverb`$_{\texttt{IV}}$



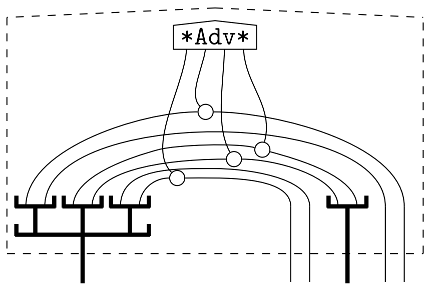
$^{-1}[[n \cdot -1n] \cdot [n]] \cdot n \cdot {}^{-1}n \cdot [n]$

`Attributive Adverb`$_{\texttt{TV}}$



$^{-1}n \cdot n \cdot {}^{-1}n \cdot [n \cdot n] \cdot n^{-1} \cdot n \cdot [[n \cdot {}^{-1}n] \cdot [n \cdot n] \cdot [n^{-1} \cdot n]]^{-1} \cdot n$

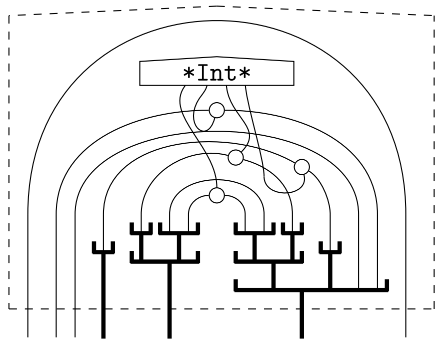
`Predicative Adverb`$_{\texttt{TV}}$



$^{-1}[[n \cdot {}^{-1}n] \cdot [n \cdot n] \cdot [n^{-1} \cdot n]] \cdot n \cdot {}^{-1}n \cdot [n \cdot n] \cdot n^{-1} \cdot n$

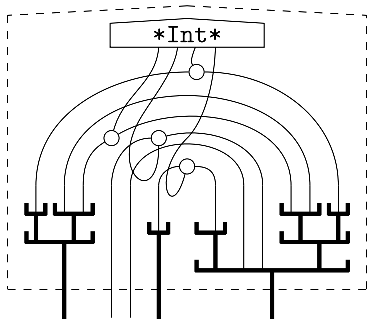
Intensifiers. Intensifiers are words like `very`, `extremely`, `terribly`, which intensify the meaning of adverbs or adjectives. By contrast, mitigators are words or phrases like `a little`, `not so`, `somewhat`, which reduce the intensity of adverbs or adjectives, but as we are just concerned with grammatical structure, we will refer to these too as intensifiers. We consider intensifiers to occur before the word they modify. We exhibit intensifiers for adverbs for intransitive verbs here.

`Intensifier`$_{\texttt{A.Adv.IV}}$



$^{-1}n \cdot n \cdot {}^{-1}n \cdot [n] \cdot [[n \cdot {}^{-1}n] \cdot [n]]^{-1} \dots$
$[n \cdot {}^{-1}n \cdot [n] \cdot [[n \cdot {}^{-1}n] \cdot [n]]^{-1}]^{-1} \cdot n$

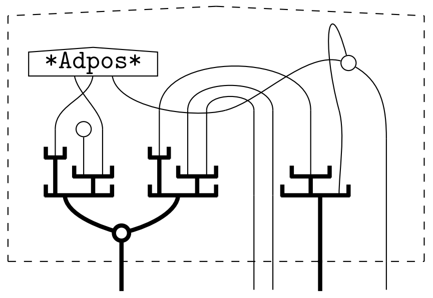
`Intensifier`$_{\texttt{P.Adv.IV}}$



$^{-1}[[n \cdot {}^{-1}n] \cdot [n]] \cdot n \cdot {}^{-1}n \cdot [n] \dots$
$[^{-1}[[n \cdot {}^{-1}n] \cdot [n]] \cdot n \cdot {}^{-1}n \cdot [n]]^{-1}$

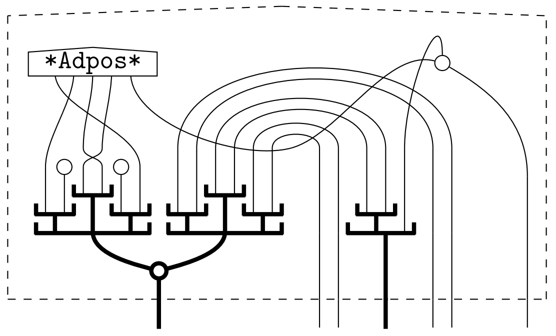
Adpositions. Adpositions relate sentences to nouns, or sentences to sentences. For example `Alice plays in the garden until Bob calls her`. Here `in` relates the phrase `Alice plays` to the noun `(the) garden`, and `until` relates `Alice plays` to the phrase `Bob calls her`. Here we only consider the phrase-noun kind, for which we have a family of internal wirings, one for each kind of verb phrase that precedes the adposition.

`Adposition`$_{\texttt{IV}}$



$^{-1}[[n \cdot {}^{-1}n] \cdot [n]] \cdot n \cdot {}^{-1}n \cdot [[n] \cdot n] \cdot n^{-1} \cdot n \cdot n^{-1}$

`Adposition`$_{\texttt{TV}}$

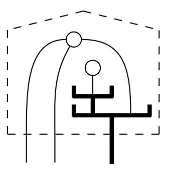


$^{-1}[[n \cdot {}^{-1}n] \cdot [n \cdot n] \cdot [n^{-1} \cdot n]] \cdot n \cdot {}^{-1}n \cdot [[n \cdot n] \cdot n] \cdot n^{-1} \cdot n \cdot n^{-1}$

6.2 Functional words

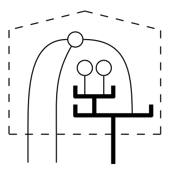
Relative Pronouns. There are subject and object relative pronouns for transitive and ditransitive verbs. Intransitive verbs only have subject relative pronouns.

`Sub. Rel. Pron.`$_{\texttt{IV}}$



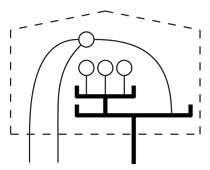
$^{-1}n \cdot n \cdot [^{-1}n \cdot [n]]^{-1}$

`Sub. Rel. Pron.`$_{\texttt{TV}}$



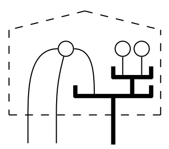
$^{-1}n \cdot n \cdot [^{-1}n \cdot [n \cdot n]]^{-1}$

`Sub. Rel. Pron.`$_{\texttt{DV}}$



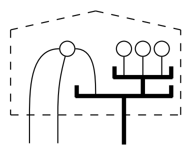
$^{-1}n \cdot n \cdot [^{-1}n \cdot [n \cdot n \cdot n]]^{-1}$

`Ob. Rel. Pron.`$_{\texttt{TV}}$



$^{-1}n \cdot n \cdot [[n \cdot n] \cdot n^{-1}]^{-1}$

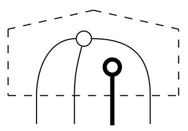
`Ob. Rel. Pron.`$_{\texttt{DV}}$



$^{-1}n \cdot n \cdot [[n \cdot n \cdot n] \cdot n^{-1}]^{-1}$

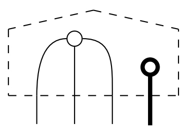
When we construct compound phrases with adpositions later, we additionally consider the following generalised relative pronouns, where the bold dot is explained in (29).

`Sub. Rel. Pron.`$_{\texttt{compound}}$



$^{-1}n \cdot n \cdot [[[\cdots]]]^{-1} \cdot n$

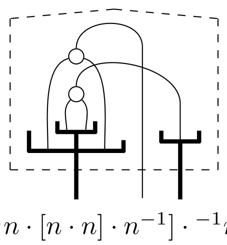
`Ob. Rel. Pron.`$_{\texttt{compound}}$



$^{-1}n \cdot n \cdot (n^{-1})^{-1} \cdot [[[\cdots]]]^{-1}$

Reflexive Pronouns. These are words ending with `-self`. To capture `snake eats itself`, we consider the reflexive pronoun `itself` to modify the transitive verb it follows, into the 'intransitive verb' `eats itself`.

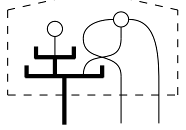
`Reflexive pronoun`



$^{-1}[^{-1}n \cdot [n \cdot n] \cdot n^{-1}] \cdot {}^{-1}n \cdot [n]$

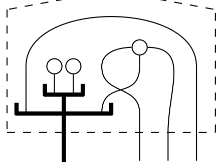
Adjectivalisation. The gerund `-ing` turns verbs into an adjective-like word. It is appropriate both for pre- or post-position while keeping the same internal wiring.

`-ing`$_{\texttt{IV}\mapsto\texttt{Pred.Adj.}}$



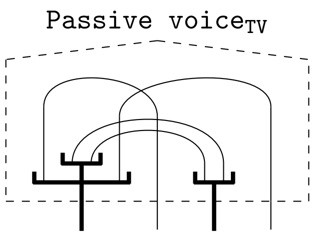
$^{-1}[^{-1}n \cdot [n]] \cdot {}^{-1}n \cdot n$

`-ing`$_{\texttt{TV}\mapsto\texttt{Pred.Adj.}}$



$^{-1}[^{-1}n \cdot [n \cdot n] \cdot n^{-1}] \cdot {}^{-1}n \cdot n \cdot n^{-1}$

Passive voice. This sentence `uses` active voice, which we have been modelling so far. Passive voice `is used in` this sentence. The passive voice 'word' (which we draw below), when placed after the verb, turns `uses` into `used`, pushing the subject to the end.



$$^{-1}[^{-1}n \cdot [n \cdot n] \cdot n^{-1}] \cdot {}^{-1}n \cdot [n \cdot n] \cdot n^{-1}$$

6.3 Examples

We now provide a number of examples of how the internal wirings proposed above, enable us to relate different grammatical constructs just as in the case of what the relative pronoun and verb internal wirings did for sentences (6) and (7).

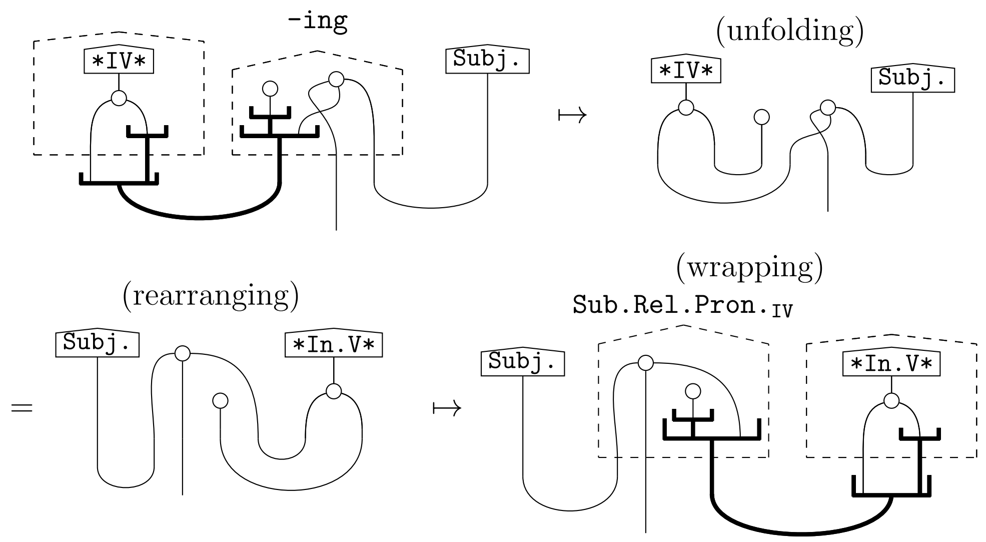
We omit the pregroup typings, instead depicting the pregroup proofs (see Definition 4.1) directly. Wrapping gadgets that occur within internal wirings correspond to words with pregroup typings that contain brackets, for example, nouns together forming a sentence type. Wrapping gadgets that occur outside internal wirings correspond to formally introducing brackets as a pregroup proof-step in a pregroup proof.

We relate:

- `Dance -ing man`

to:

- `Man that dances`



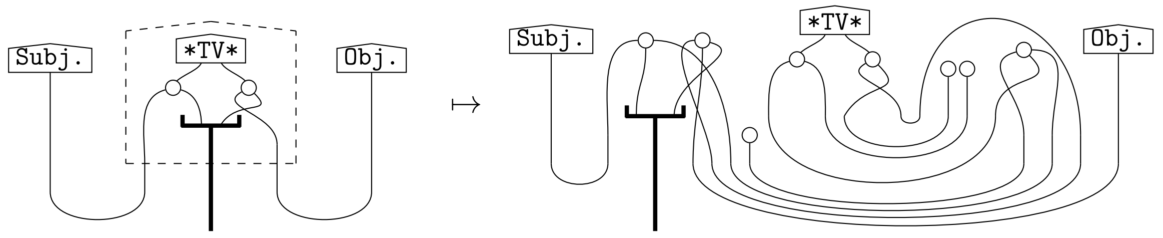
We relate:

- `Farmer sows corn`

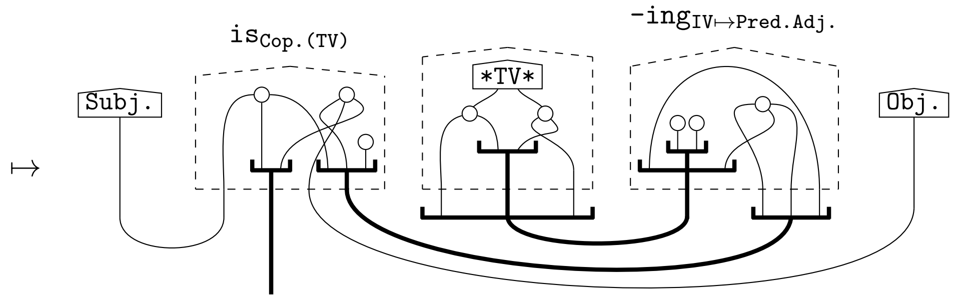
to:

- `Farmer is sow -ing corn`

(unfolding, adding spider units, rearranging)



(wrapping, recovering pregroup proof with bracketing)

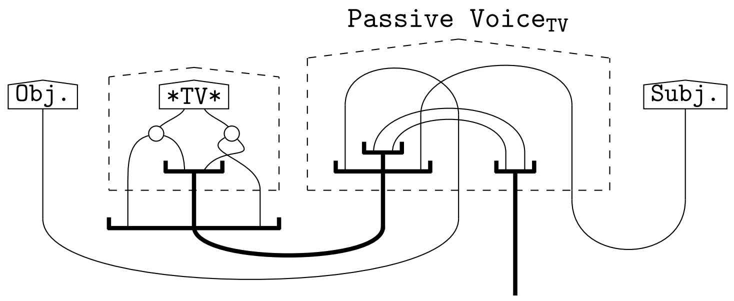


We relate:

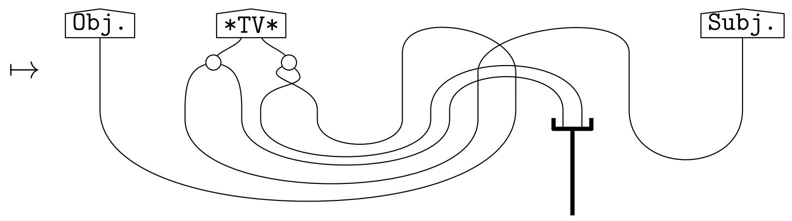
- `Alice` $\overbrace{\texttt{is bored by}}^{\text{passive voice}}$ `the class`

to:

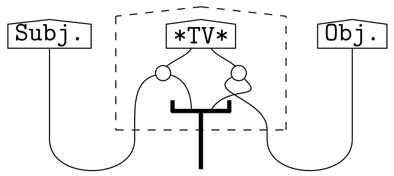
- `The class bores Alice`



(unfolding)



(rearranging wires, wrapping)

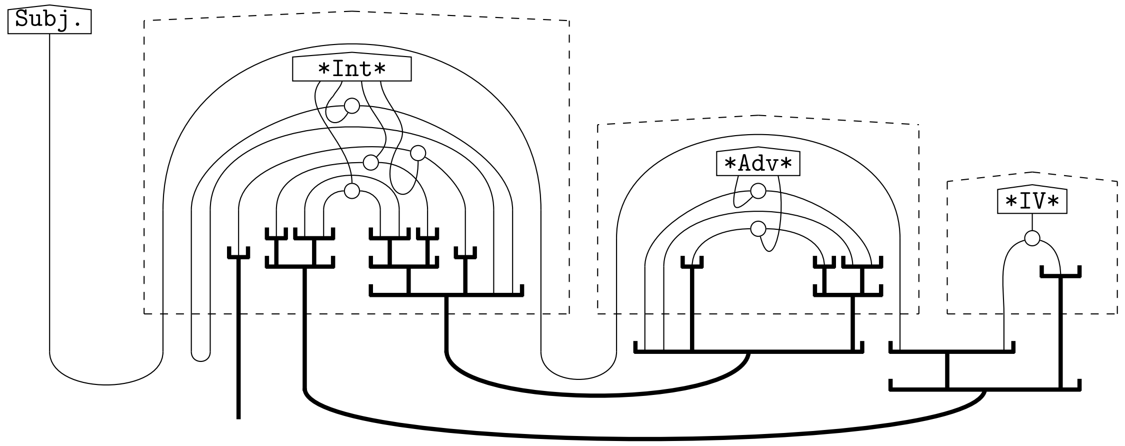
$\mapsto$ 

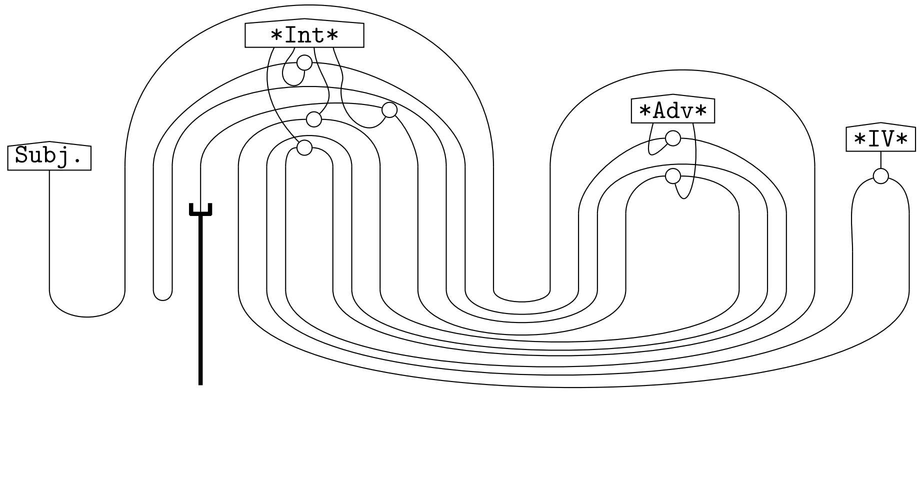
We relate:

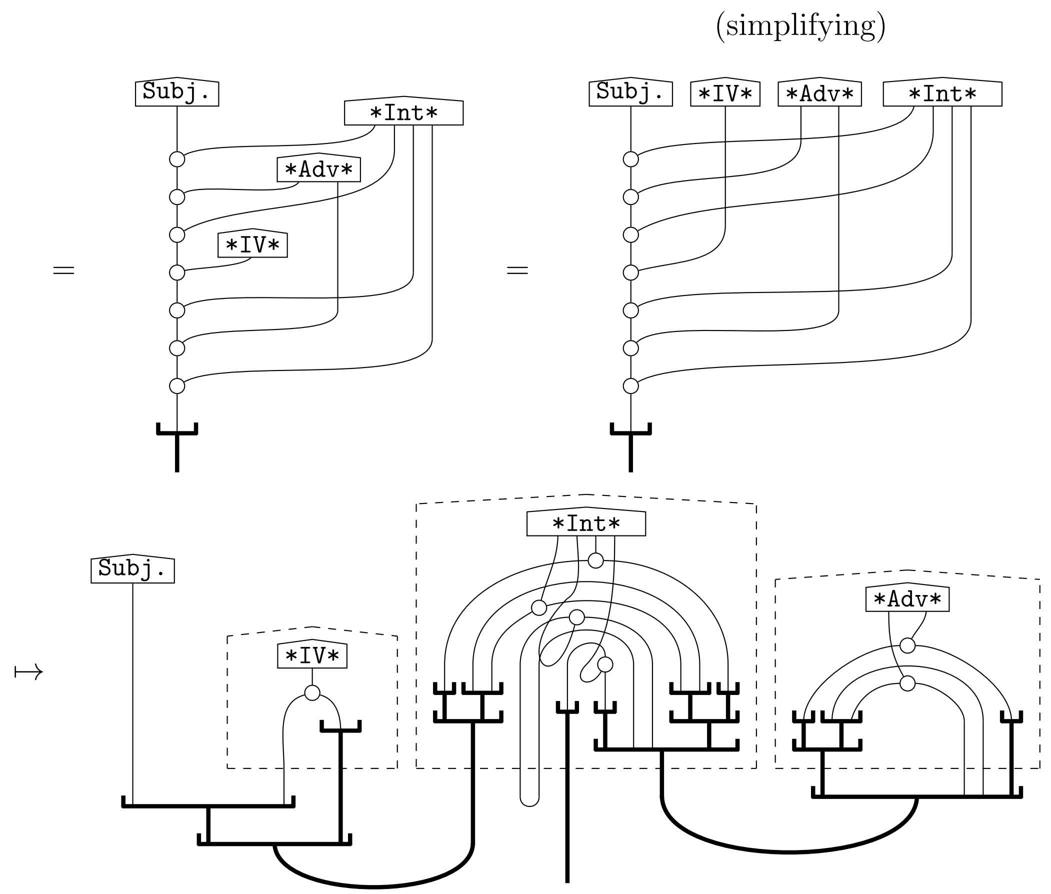
- `Alice runs very quickly`

to:

- `Alice very quickly runs`



$\mapsto$ 

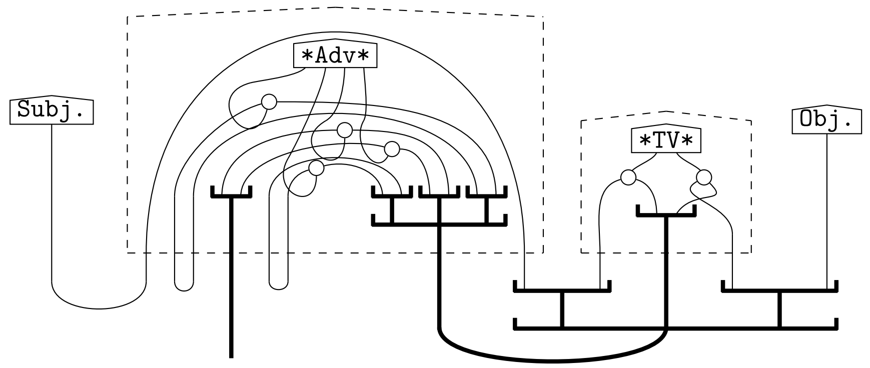


We relate:

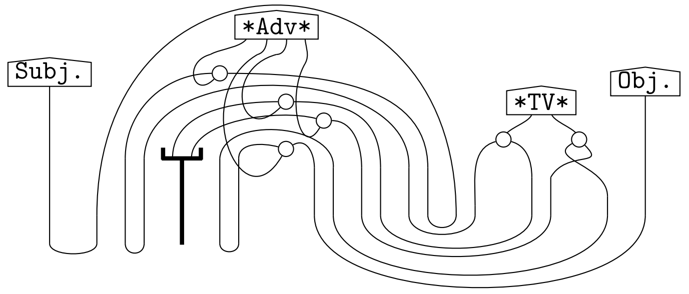
- `Alice washes Fido gently`

to:

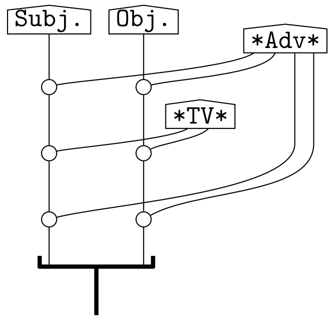
- `Alice gently washes Fido`



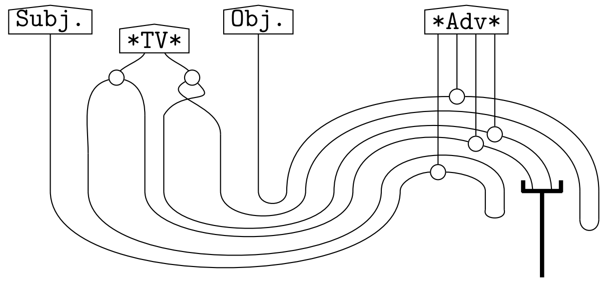
$\mapsto$



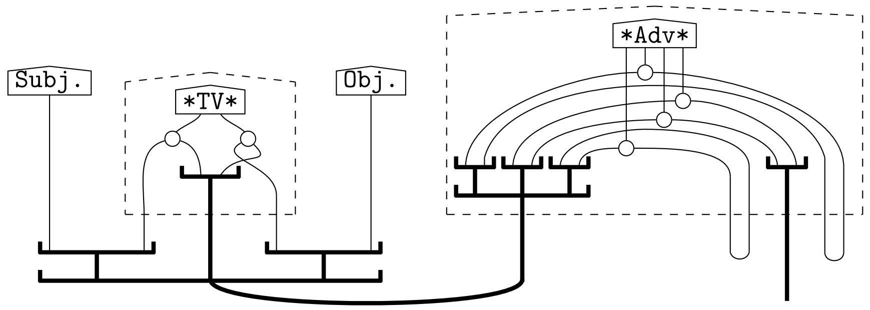
$=$



$=$



$\mapsto$



6.4 Further derived internal wirings

From the internal wirings above, we can derive some more, using the fact that we expect certain composites to be grammatically equal.

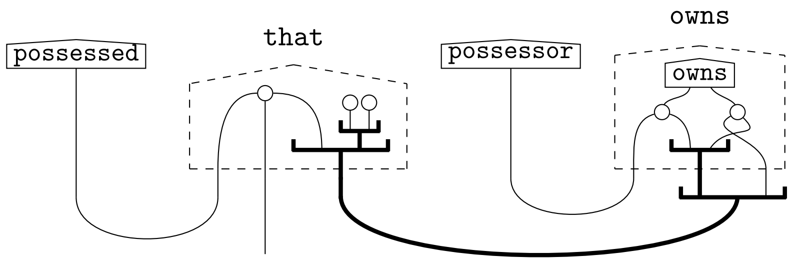
From:

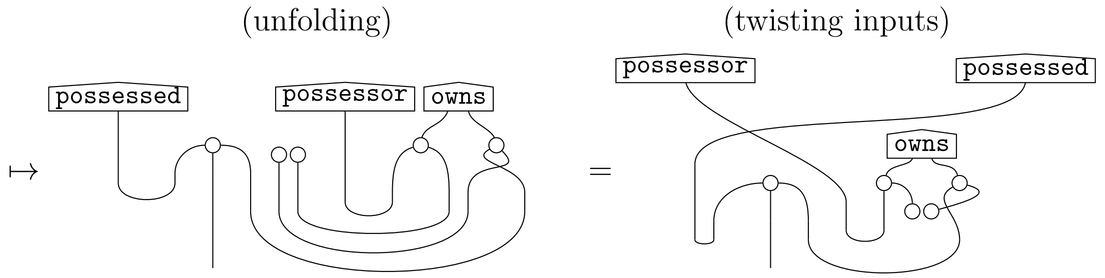
- `(possessed) that (possessor) owns`

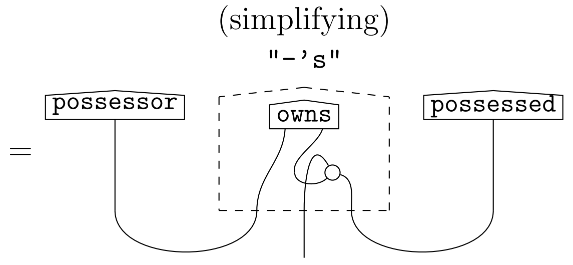
we derive the possessive modifier:

- `(possessor)` <u>`’s`</u> `(possessed)`

up to rearranging the order of noun-arguments. So we consider possessive pronouns such as his to modify the nominative he with -'s.





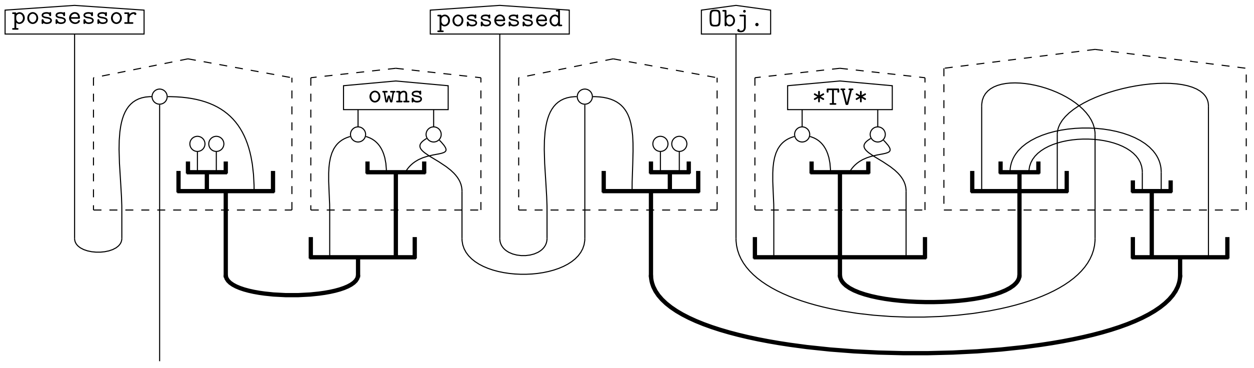


From:

- author that owns book that John (was) entertain(s) -ed (by)

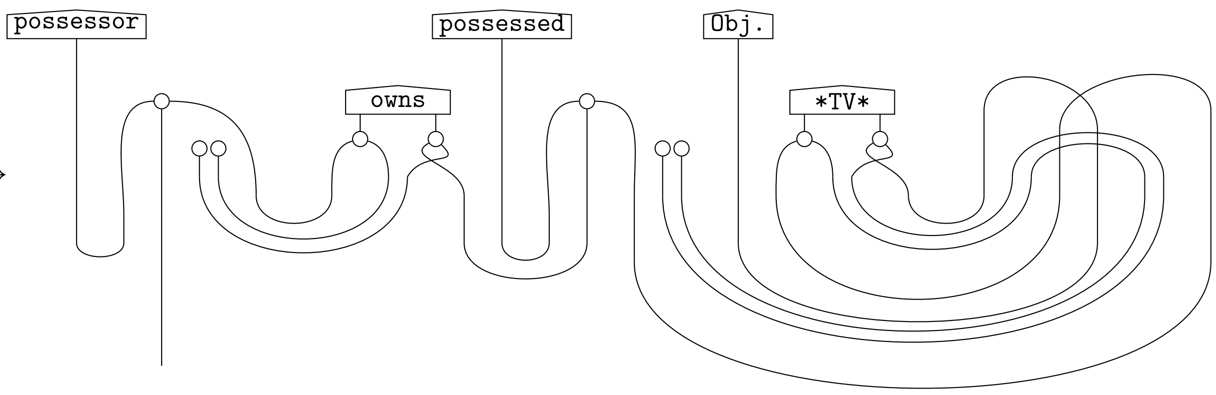
we derive a possessive relative pronoun:

- author <u>whose</u> book entertained John



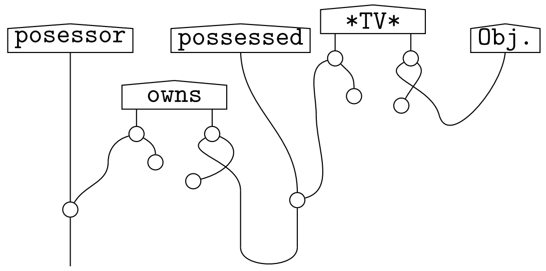
(unwrapping wires)

$\mapsto$



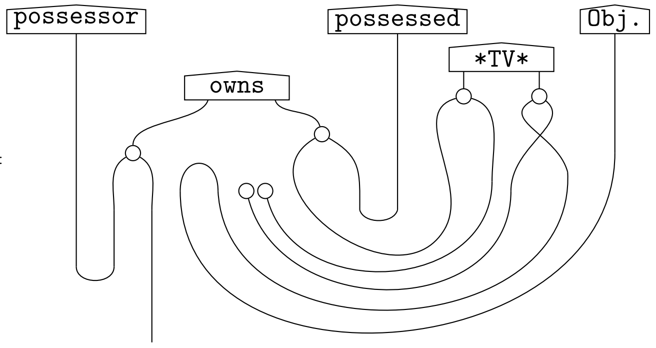
(simplifying)

$=$



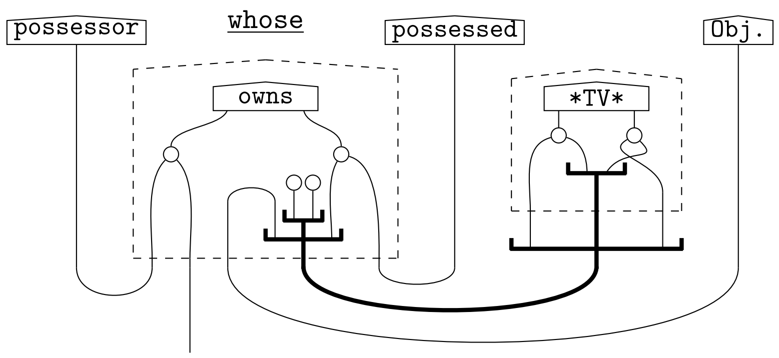
(dragging wires into place)

$=$



(recovering a pregroup proof with bracketing)

$\mapsto$



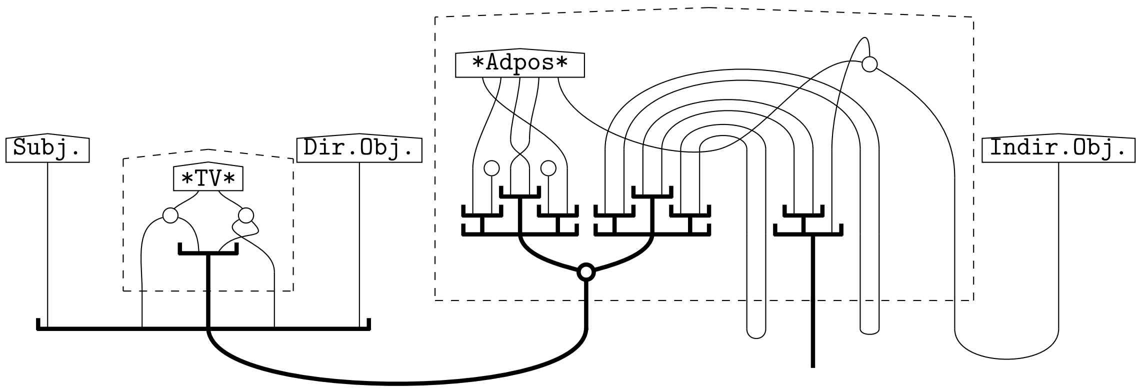
From transitive verbs and adpositions:

- `Bob gives`$_{\texttt{TV}}$ `flowers to Claire`

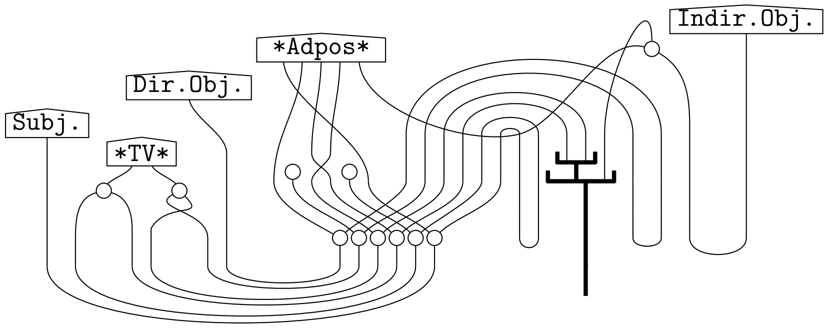
we derive ditransitive verbs:

- `Bob gives`$_{\texttt{DV}}$ `Claire flowers`

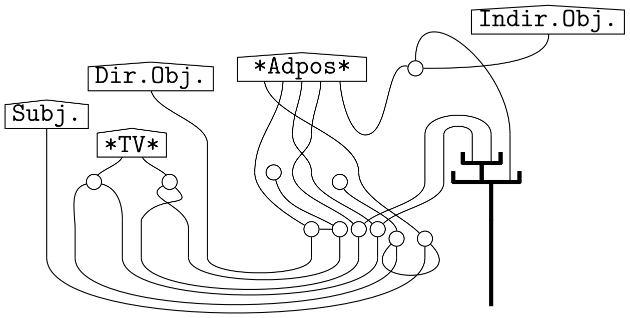
Phrase-noun adpositions indeed turn k-ary verb phrases into $k+1$-ary verb phrases, hence we can deconstruct ditransitive verbs using transitive verbs and phrase-noun adpositions.



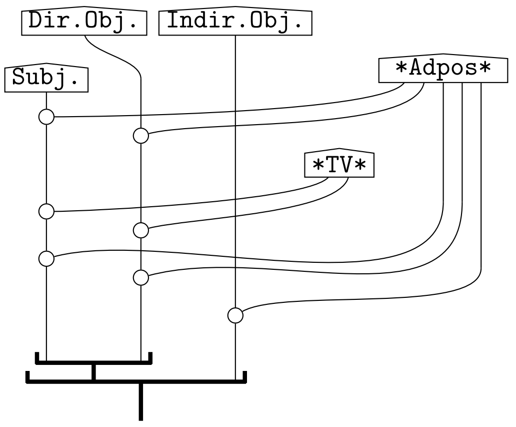
$\mapsto$



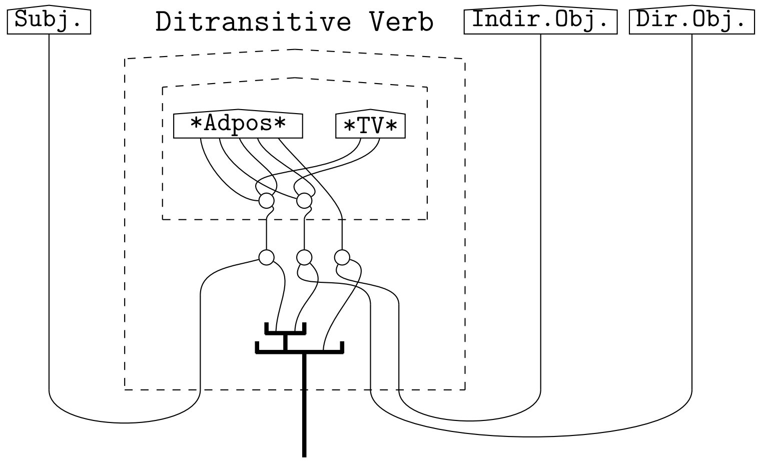
$=$



$=$



$\mapsto$



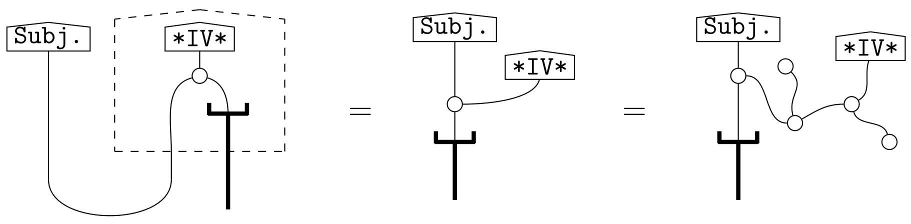
From:

- `Alice swims`

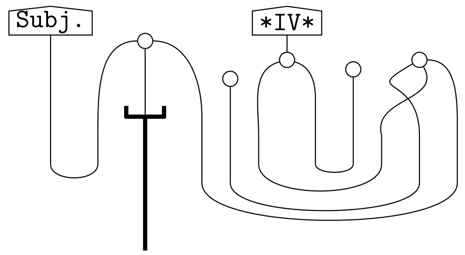
we derive the copula `is` **in:**[7]

- `Alice is swim -ing`

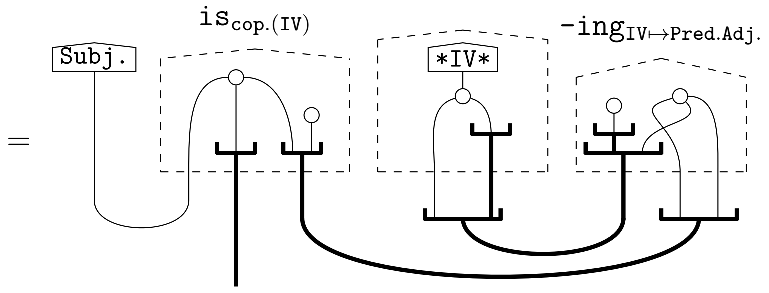
(un-fusing spiders)



$=$



[7]The equivalence holds for present tense declaratives.

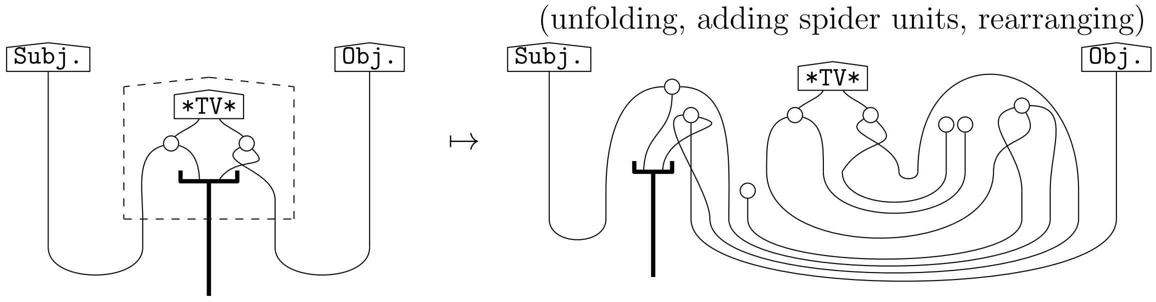


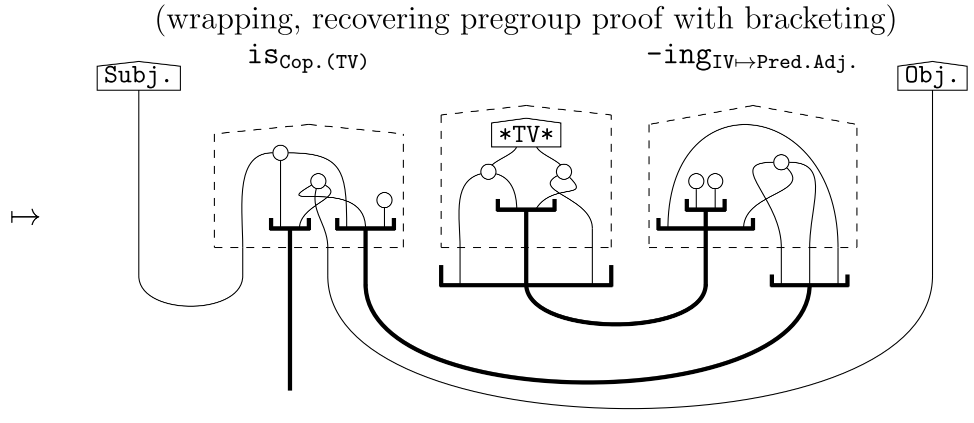
Similarly:

- `farmer sows corn`

is the same as:

- `farmer is sow -ing corn`




(wrapping, recovering pregroup proof with bracketing)
↦
is$_{\text{Cop.(TV)}}$
-ing$_{\text{IV}\mapsto\text{Pred.Adj.}}$
Subj.
TV
Obj.

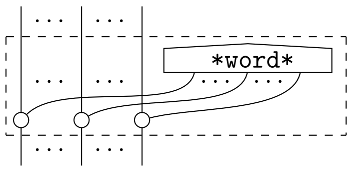
7 Pregroup diagrams to language circuit algorithm

We first define language circuits, and then we present the algorithm that turns pregroup diagrams of grammatical sentences into language circuits, and apply it to examples.

7.1 Language circuits

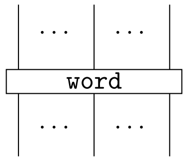
We now formally define language circuits. We first generalise the gates of (24) and combs of (28) to arbitrary words.

Definition 7.1. A basic gate is a gate with the following internal structure:



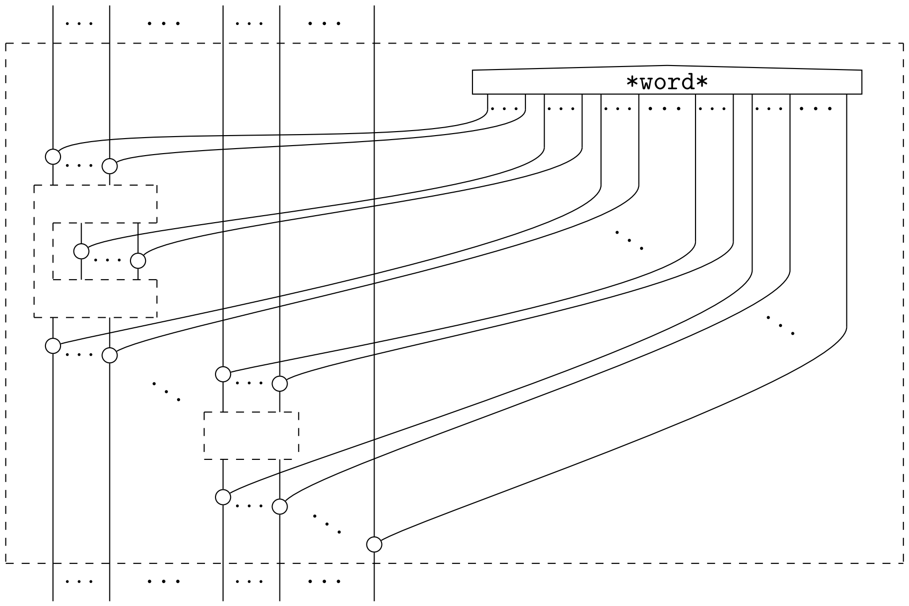
(31)

By Lemma 4.8, without loss of generality, they represent general gates:



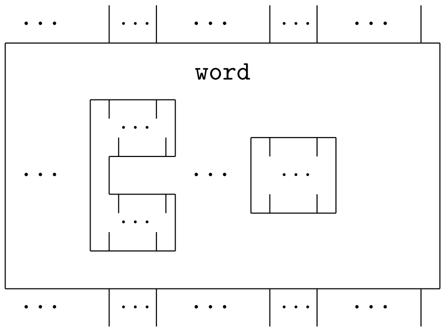
(32)

A higher gate is a gate that acts on some combination of wires, basic gates, and other higher gates. Where the internal dotted boxes indicate gaps to place other gates, higher gates have the following internal structure:



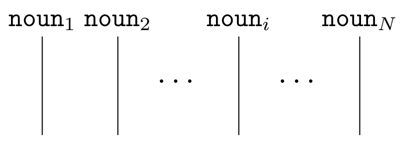
(33)

By Lemma 4.11, without loss of generality, they represent general combs:



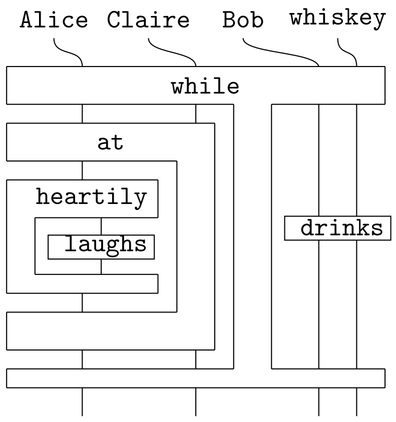
(34)

Definition 7.2. A language gate is a gate made up of basic and higher gates, with no gaps. A language circuit has vertical wires labeled by nouns:



and the gates in it are language gates.

Here's an example of a language circuit (not in spider-form):

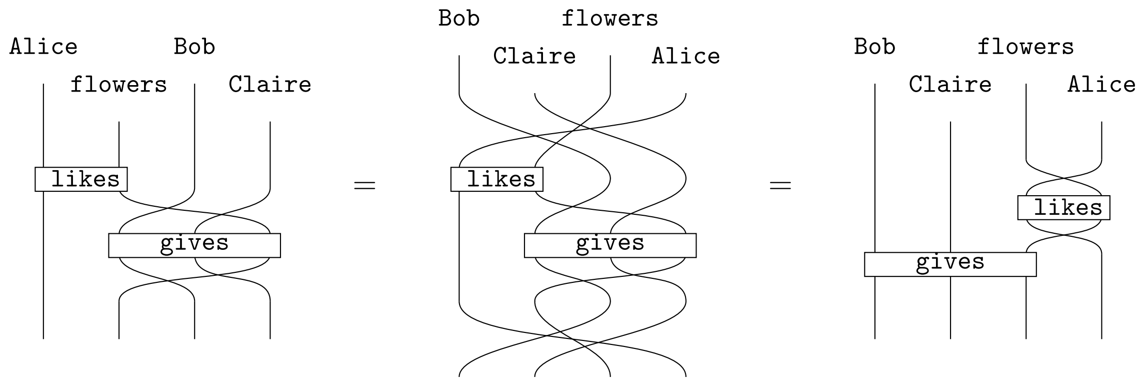


(35)

`Alice heartily laughs at Claire while Bob drinks whiskey`

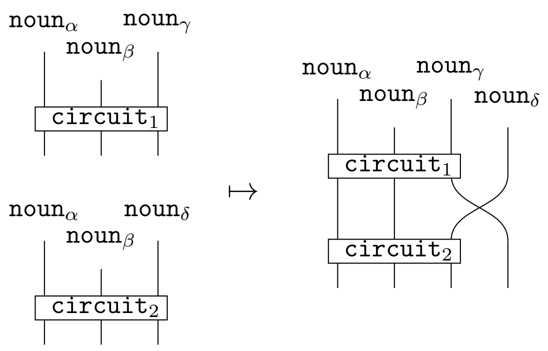
The algorithm that we present below will produce either a basic or higher gate for every content word in the sentence. Well-formed subphrases will correspond to language gates.

Order-freeness of noun-wires. We consider two language circuits to be the same if their connectivity is the same. So rearranging vertical wires does not change a circuit:



This is why constructions that rearrange noun order such as the passive voice and the possessive modifier have internal wirings that contain twists, while the diagrams for the constructed and original sentences stay equal up to spider-fusion, wrapping, and unfolding. All three of these operations conserve connectivity.

Composing Circuits. We compose circuits obtained from sentences in text in the obvious way, connecting wires of matching nouns.



Resolving pronouns. The <u>referent</u> is the noun that a pronoun refers to, for example, in the sentence:

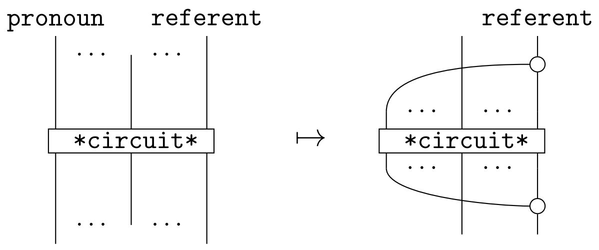
`Bob drinks beer that he bought`

the noun `Bob` is the referent of the pronoun `he`.

<u>Resolving</u> a pronoun means to determine its referent, especially when there is ambiguity, for example, in the text:

`Alice goes to dinner with Claire; she pays the bill`

the pronoun `she` can refer to either `Alice` or `Claire`. We do not provide an account here of how to determine what a pronoun refers to, but any such procedure (e.g. [39, 11]) is modularly compatible with our approach. For language circuits, if a pronoun gets resolved we fuse its wire with the wire of its referent:



7.2 The Algorithm

We want to turn a grammatical sentence into a language circuit. Here we first list the main steps for getting from a pregroup diagram with wrapping to a language circuit:

1. Replace words with their corresponding internal wirings from Section 6.
2. Replace sentence wires with appropriate wrapped wires.
3. Delete the output wire.
4. Replace each noun (including pronouns) with a copy-spider. The two free legs will become the vertical wires in the circuit, labelled by the noun.

5. Unwrap all wrapped wires using Section 5.2.

6. Resolve pronouns as described above.

7. Use spider-fusion to obtain a language circuit.

The only missing step is turning a sentence into a pregroup diagram with wrapping, for which there are standard tools available. We now state the full algorithm in pseudo-code:

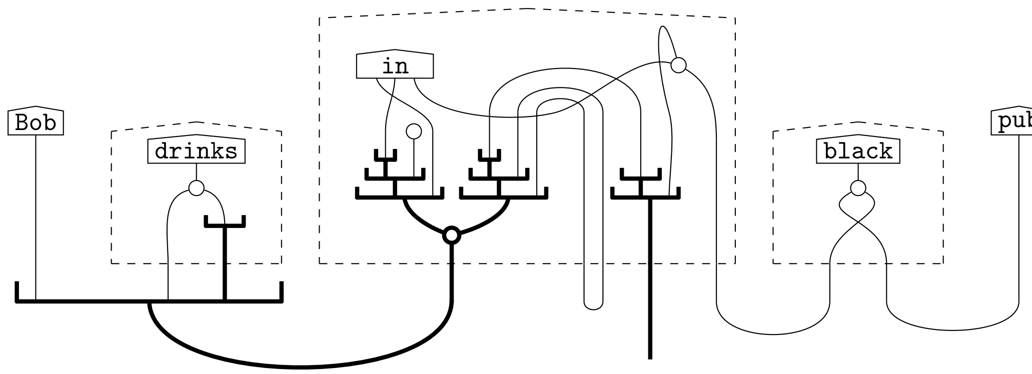
- Let S be a sentence in the fragment of English we have described, as a list of words.
- Let ***Parse*** denote a program that takes sentences in the fragment and returns their pregroup diagrams with wrapping.
- Let `NOUNS` be the set of nouns in the ambient language, of which `PRONOUNS` is a subset.
- Let ***PrnRes*** denote a pronoun resolution oracle, which takes a pair (p, n), $p \in \texttt{PRONOUNS}, n \in \texttt{NOUNS} - \texttt{PRONOUNS}$ and returns `True` just when the pronoun p refers to n, and `False` otherwise.

Algorithm 1: Grammar to Circuit Algorithm

```
Input: A sentence $S$
Result: A language circuit
Data: Parse, PrnRes, NOUNS, PRONOUNS
BASICWIRES $\leftarrow \{w \in S \cap$ NOUNS$\}$ ;            // 'BASICWIRES' is nouns of $S$
DIAGRAM $\leftarrow$ Parse($S$) ;    // 'DIAGRAM' is pregroup diagram with wrapping
if 'DIAGRAM' has single output wire o then
    Append appropriate deletion to o in DIAGRAM ;        // delete output wire
    for $w \in$ BASICWIRES do
        Erase $w$'s box from DIAGRAM, creating open wire ;             // copy nouns
        Append rightmost output of a copy-spider to the open wire;
        Pull free input of copy-spider to the top of DIAGRAM;
        Pull free output of copy-spider to the bottom of DIAGRAM;
        Label this new wire with $w$;
    end
    while There remain wrapped wires in DIAGRAM do
        Apply unfolding from Definition 5.2 ;                          // unwrap
    end
    for $w_1, w_2 \in$ BASICWIRES do
        if $w_1 \in$ PRONOUNS and $w_2 \notin$ PRONOUNS and PrnRes($w_1, w_2$) then
            Attach outputs of a copy-spider to $w_1$, $w_2$ inputs;
            Attach inputs of upside-down copy-spider to $w_1$, $w_2$ outputs;
            relabel merged wire as $w_2$ ;                    // resolve pronouns
        end
    end
    while There remain deletions, cups, or caps of non-wrapped wires do
        Use spider-fusion to remove deletions, cups, and caps;
    end
    Simplify to obtain language circuit ;                            // clean up
    return DIAGRAM ;                                  // return language circuit
else
    return False ;                          // (Fail if input ungrammatical)
end
```

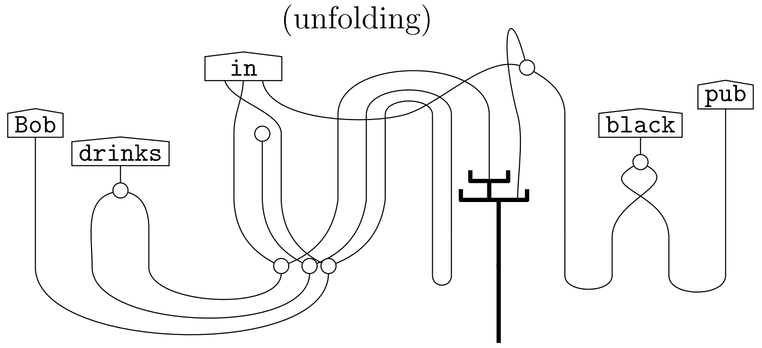
7.3 Examples

`Bob drinks in (the) black pub.`



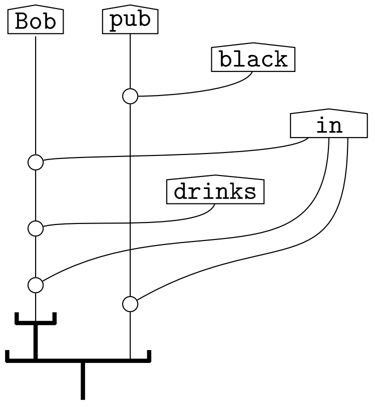
(unfolding)

$\mapsto$

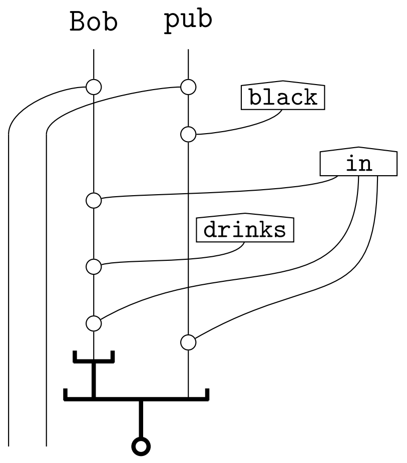


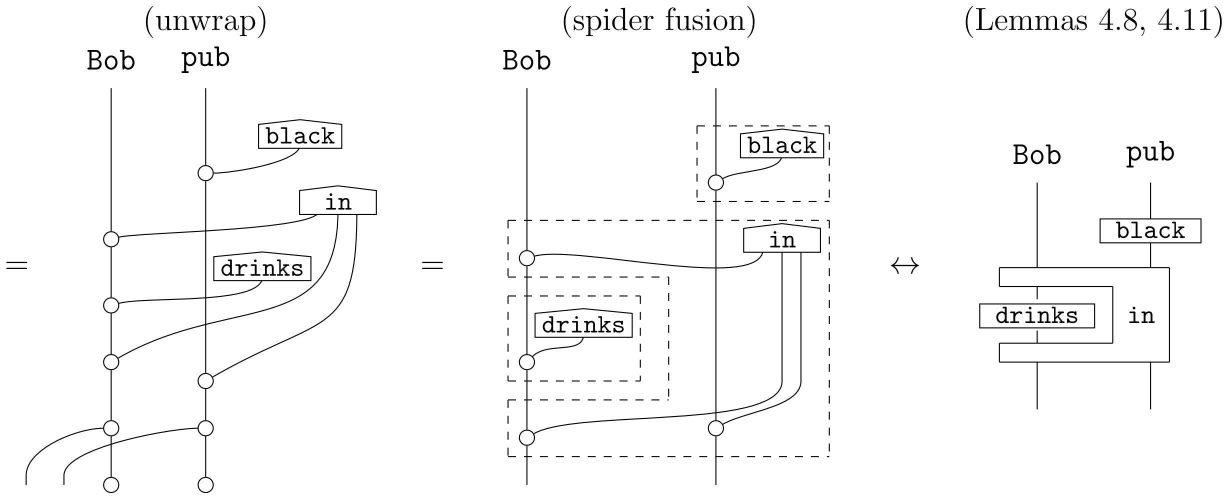
(delete outputs, copy nouns)

$=$



$\mapsto$



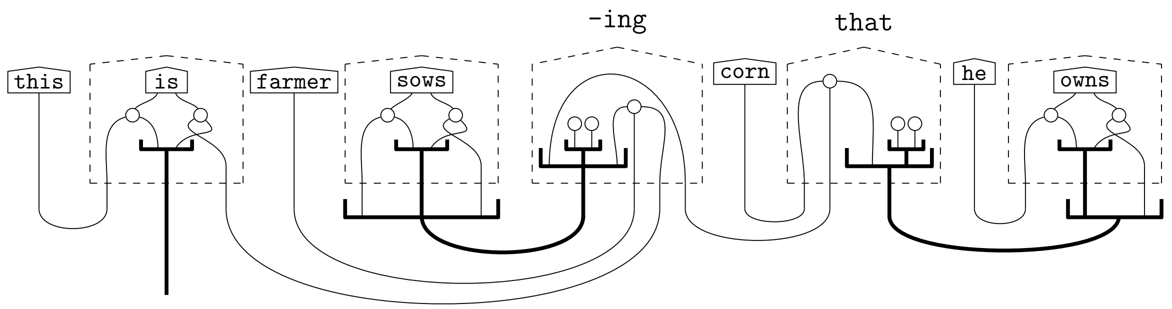


`This is the farmer sowing his corn.`

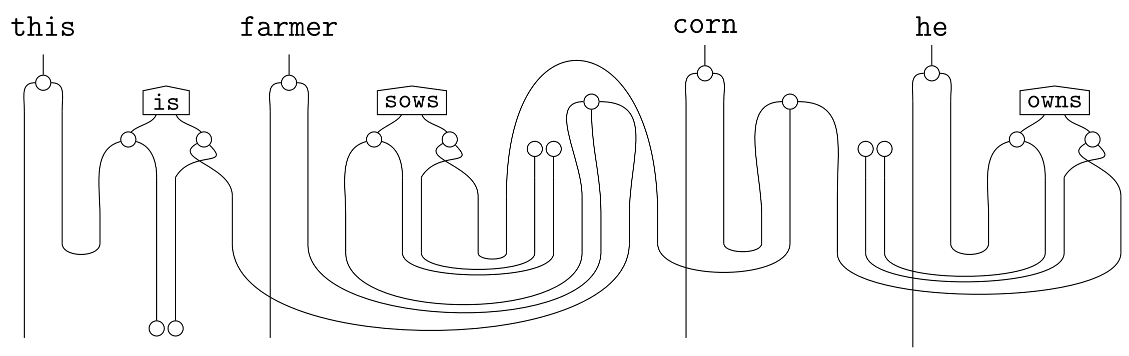
We treat `is` as a regular transitive verb, and following our derivation of the possessive pronoun, we treat `his corn` as `corn that he owns`. The catalog has enough for us to express the full sentence as:

`This is (the) farmer sow-ing corn that he owns.`

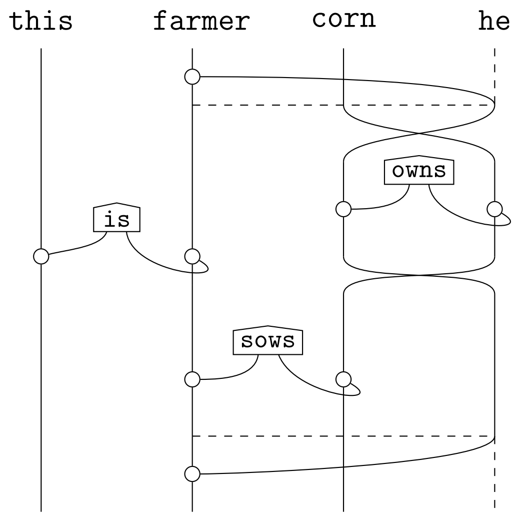
Here is the pregroup diagram with wrapping we obtain:



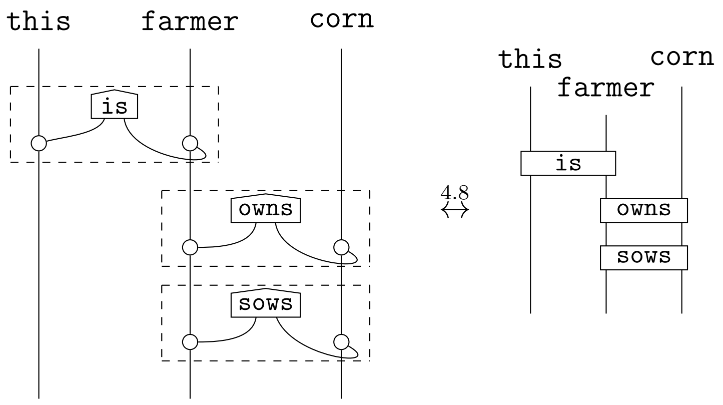
After we delete the output wire, replace the noun-boxes by copy-spiders, and unfold all the wires, we are left with only noun wires, spiders, and word-boxes:



Now we use spider-fusion to eliminate cups, caps, and deletes. We move each word-box to its own horizontal level, and bring its connected spiders to that level. By construction, at this point we have word gates. Now we resolve the pronoun `he` by merging that wire with `farmer` using spiders (see the dotted lines), and keeping only the `farmer` label. The pronoun `this` has no referent, so we leave it alone:



We can continue to spider-fuse to completely eliminate the `he` wire. When there are no pronouns left to resolve, we can apply Lemma 4.8 to simplify presentation and finish:

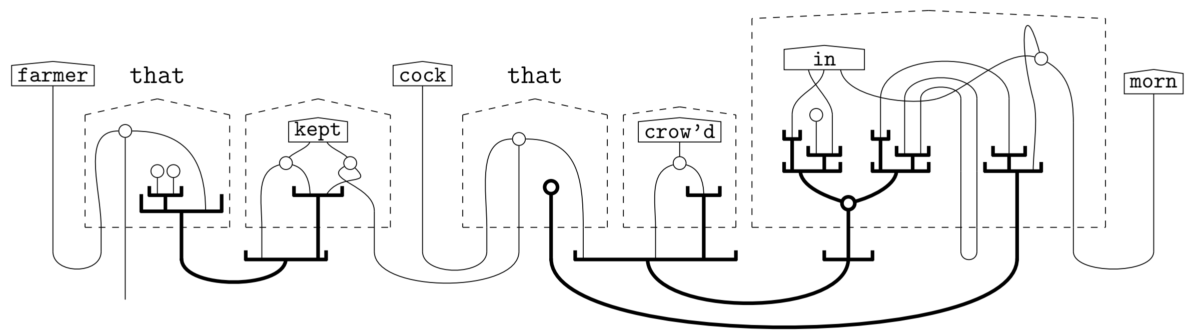


`...(the farmer) kept the cock that crow’d in the morn.`

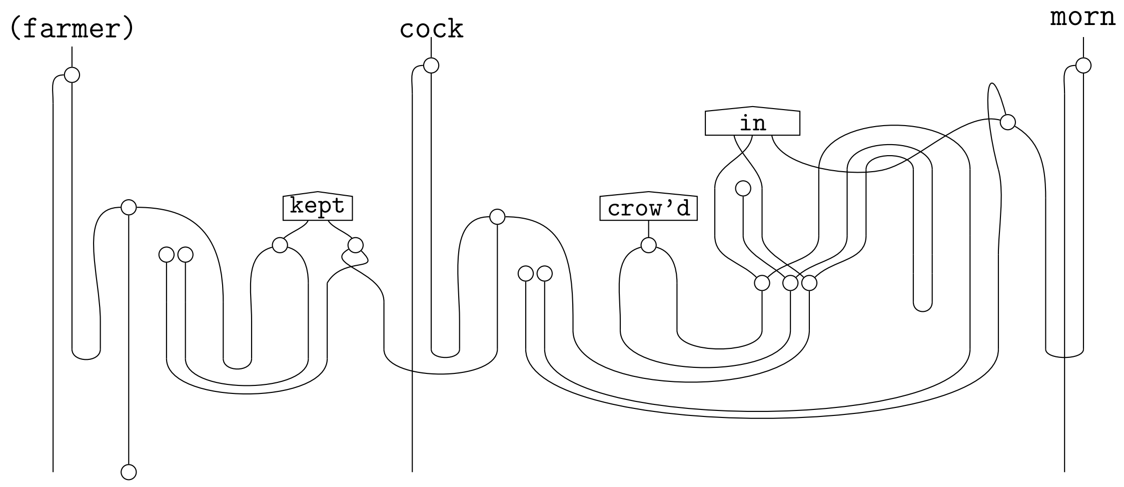
We can also apply the algorithm to noun phrases. After we obtain a language circuit from a noun-phrase this way, we can compose it with other circuits. So now, we build on the sentence we just translated, adding on:

`(the farmer) that kept (the) cock that crow’d in (the) morn.`

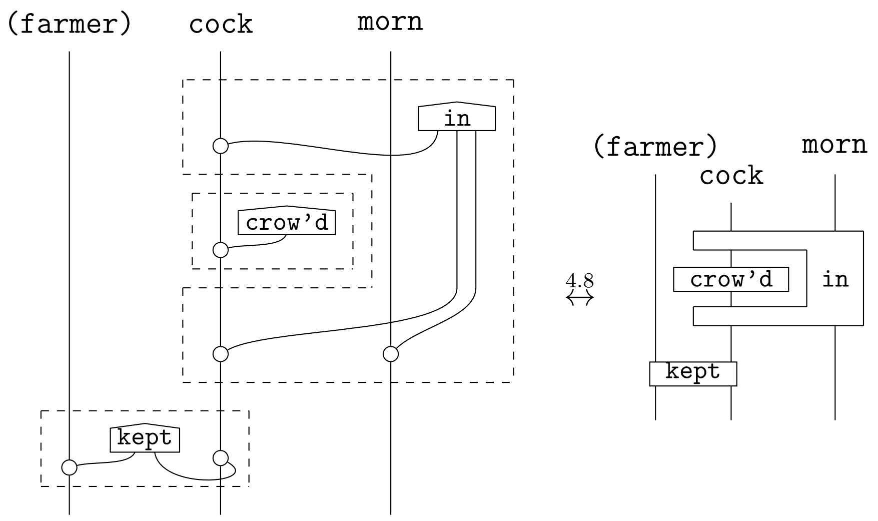
We treat `the cock` and `the morn` as nouns, and `crow’d` as an intransitive verb. Once we fill in internal structures from our catalog, we get:



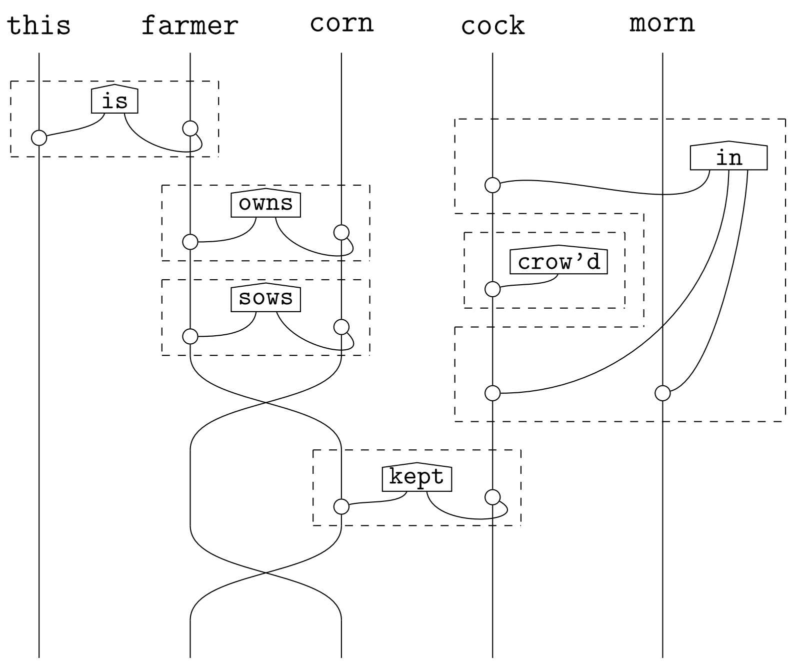
We replace noun boxes with copies, delete the output wire, and unfold. We label `farmer` in brackets to remember that we have to connect it to the farmer mentioned earlier:



Now we can use spider-fusion to simplify, obtaining a language circuit:



Now we can put the two language circuits we have obtained together. When we do so, we twist some wires to keep the order of input wires the same as the order of output wires, and we try to keep gates from crossing over other wires. We obtain a big, composite language circuit, which we depict in full:



A nursery rhyme: `The house that jack built.` Here is the final stanza of the nursery rhyme, which is one very long sentence:

```
This is the farmer sowing his corn
That kept the cock that crow'd in the morn
That waked the priest all shaven and shorn
That married the man all tatter'd and torn
That kissed the maiden all forlorn
That milk'd the cow with the crumpled horn
That tossed the dog
That worried the cat
That killed the rat
That ate the malt
That lay in the house that Jack built
```

To turn this long sentence into a circuit, we make some concessions:

- We overlook determiners and tenses.
- We gloss some phrases with close equivalents, for example:

  `...man all tattr'd and torn...` $\mapsto$ `...tattr'd torn man...`

- We re-express certain words using our catalog, for example:

  `...his corn...` $\mapsto$ `...corn that he owns...`

- We resolve pronouns as follows:

  `farmer sowing corn that he owns` $\mapsto$ `farmer sowing corn that (farmer) owns`

- We break up the sentence into smaller ones, for example:

  `farmer kept cock that crow'd in morn`

  becomes two sentences:

  `farmer kept cock`

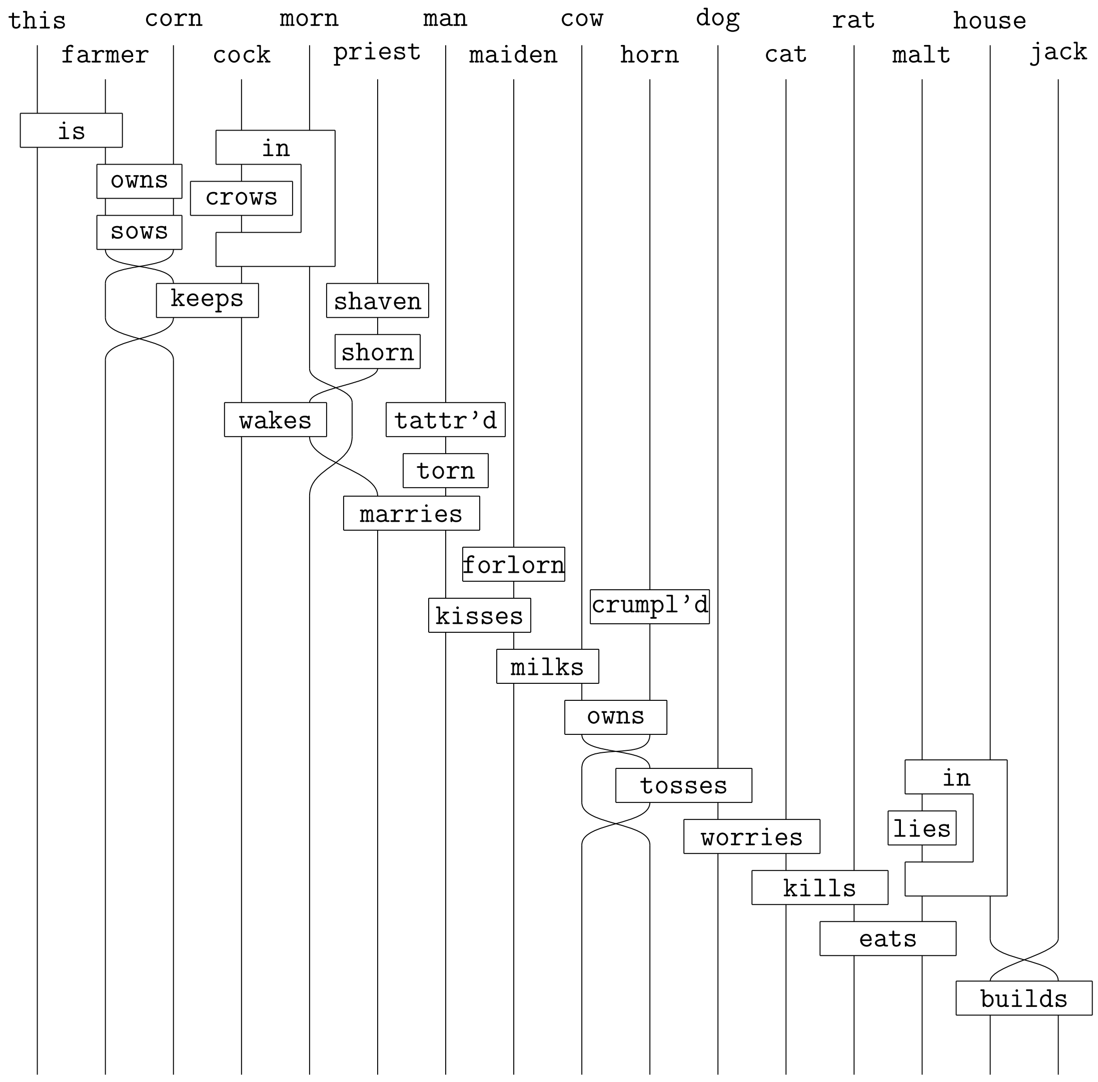
  `(cock) crow'd in morn`

  The circuit we obtain after composition is the same in both cases, as in (3).

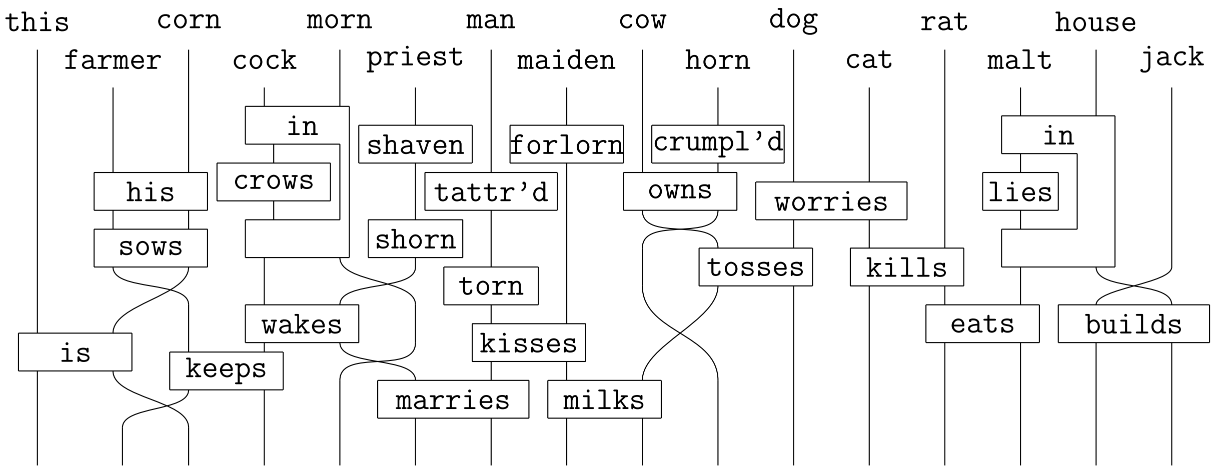
After all this, the long sentence becomes the following text:

```
This is farmer sows -ing corn;
(farmer) owns corn;
(farmer) keeps cock;
(cock) crows in morn;
(cock) wakes shaven shorn priest;
(priest) marries tatter'd torn man;
(man) kisses forlorn maiden;
(maiden) milks cow;
(cow) owns crumpled horn;
(cow) tosses dog;
(dog) worries cat;
(cat) kills rat;
(rat) eats malt;
(malt) lies in house;
(house) Jack builds -ed
```

We see nouns, intransitive and transitive verbs, `-ing`, the passive voice `-ed`, attributive adjectives, and adpositions: all covered by our catalog. The pregroup diagram of the original sentence is too wide to reasonably depict. However, the language circuit we obtain is exactly as one expects:



When the gates we choose to model words with slide past each other, we can greatly simplify the situation. Maybe computers will have it easier than us:



8 Related Work

The replacement of the sentence type with tensored wires used in this paper was prefigured as 'Cartesian Verbs' in [21] Sec. 2.3, which in turn had precursors in [28, 34, 33].

As circuits are naturally compositional, we immediately permit a move from the semantics of sentences to a semantics of text, as proposed in [17]. The rules governing composition of sentences in text naturally evoke *context*. Textual context has figured in DisCoCat-related papers before [48, 59], although no sentence composition mechanism was proposed, due to a (literal) bottleneck in the geometry of pregroup diagrams.

Also within the context of DisCoCat, the work by Toumi et al. [18, 57] models intersentential interaction through *discourse representation structures* [31], which however came at the cost of reducing text meaning to a scalar. Language circuits benefit from discourse representation structures without a concession in expressiveness of meaning. The circuits we introduce here can be viewed *directly* as discourse representation structures where pronouns are represented as noun-wires with undetermined label, to be connected to properly labelled wires elsewhere. A natural direction for further development of language circuits in line with discourse representation is to appropriately enrich the ambient categories so that indefinite pronouns such as `everybody` and `something` can be handled.

We remark that the explicit bracketing structure in types, which show up as the wrapping gadget diagrammatically, happens to coincide with the type structure required in Muskens' work [47] that grants discourse representation structures to pregroup grammars in generality. But it is worth emphasising that just as the jump from pregroup grammars to pregroup diagrams permits compositional semantics according to grammar, a similar compositional semantics is gained here.

The compositional semantics gained here is not without effort. The internal structures for grammatical words chosen here strictly generalise the work of [51] and [52], but were engineered in a similar fashion: following a pre-formal and human understanding of how grammatical words affect information flow and connectivity in sentences. This effort pays off. As we have seen, the passage from pregroup proofs to diagrams, and the correspondence between pregroup diagrams and pregroup proofs, treats diagrams with properly chosen grammar-spiders as a medium for meaning-preserving rewriting, which is precisely the business of Harris and Chomsky's *transformational grammar* [50]. So, where the internal structure for grammatical words are well chosen and coherent with respect to each other, the pregroup diagrams shown here before simplification into circuits can constitute an account of what Chomsky's *deep structure* [8] really is.

The language circuit formalism does not appear to detract or exclude the use of methods within traditional research in natural language syntax and semantics. In *dynamic semantics* [30, 58] models sentence meanings as I/O-transitions and text as compositions thereof. However, the approach is still rooted in predicate logic, just as Montague semantics is, hence not accounting for more general meaning spaces, and so not explicitly admitting the type structure of diagrams/monoidal categories. Dynamic semantics is a precursor of *dynamic epistemic logic (DEL)* [6, 5]; we expect that DEL, and generalisations thereof, may in fact emerge from our model of language meaning by considering an epistemics-oriented subset of meanings. In [53], static and dynamic vector meanings are explicitly distinguished, taking inspiration for the latter from dynamic semantics. There are many other logic-oriented approaches to text e.g. [3], of text organisation e.g. [42], and of the use of categorical structure.

There is a clear scope for combining language circuits with ML-methods. Some work in this direction, for the case of DisCoCat, is [40]. There is also the concrete problem of learning gate representations given diagrammatic constraints, which appears well suited for an ML approach. For instance, to model collections of spatial adpositions such as `above`, taken to be a binary gate with special properties, such as idempotence and transitivity, learning appropriate gate representations permits the resulting circuits to express and solve inference problems, such as in [26].

The fragment of English we have modelled here is a small *controlled natural language* [35, 62, 10], examples of which have found use in industry, critical machine applications, and to facilitate human-to-human communication. Accordingly, there is potential value in developing procedures to translate natural language into, semantically close equivalents in the controlled fragment, again where an ML-approach may find application. Alternatively, the expressivity of the fragment can be expanded, where the natural direction of expansion is to capture grammatical features common to multiple languages.

9 Acknowledgements

We thanks Dimitri Kartsaklis, Konstantinos Meichanetzidis, Anna Pierson, Robin Lorentz, Alexis Toumi, Giovanni de Felice, Ian Fan, Richie Yeung and the SEMSPACE referees for useful feed-back.

References

[1] S. Abramsky. Retracing some paths in process algebra. In *CONCUR'96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 1996.

[2] K. Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1937.

[3] N. Asher and A. Lascarides. *Logics of conversation.* Cambridge University Press, 2003.

[4] R. Bacon. Summa grammatica. 1245.

[5] A. Baltag, B. Coecke, and M. Sadrzadeh. Algebra and sequent calculus for epistemic actions. *Electronic Notes in Theoretical Computer Science*, 126:27–52, 2005.

[6] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, pages 43–56. Morgan Kaufmann Publishers Inc., 1998.

[7] Y. Bar-Hillel. A quasiarithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[8] G. Bedell. The arguments about deep structure. *Language*, 50(3):423–445, 1974. Publisher: Linguistic Society of America.

[9] Emily M. Bender. Linguistic typology in natural language processing. *Linguistic Typology*, 20(3):645–660, 2016.

[10] A. Bernth. Easy English: Addressing structural ambiguity. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup*, AMTA '98, pages 164–173, Berlin, Heidelberg, October 1998. Springer-Verlag.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[12] A. Carboni and R. F. C. Walters. Cartesian bicategories I. *Journal of Pure and Applied Algebra*, 49:11–32, 1987.

[13] N. Chomsky. Aspects of the theory of syntax. 1965.

[14] S. Clark, B. Coecke, E. Grefenstette, S. Pulman, and M. Sadrzadeh. A quantum teleportation inspired algorithm produces sentence meaning from word meaning and grammatical structure. *Malaysian Journal of Mathematical Sciences*, 8:15–25, 2014. arXiv:1305.0556.

[15] S. Clark and J. R. Curran. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552, 2007.

[16] B. Coecke. An alternative Gospel of structure: order, composition, processes. In C. Heunen, M. Sadrzadeh, and E. Grefenstette, editors, *Quantum Physics and Linguistics. A Compositional, Diagrammatic Discourse*, pages 1 – 22. Oxford University Press, 2013. arXiv:1307.4038.

[17] B. Coecke. The mathematics of text structure, 2019. arXiv:1904.03478.

[18] B. Coecke, G. De Felice, D. Marsden, and A. Toumi. Towards compositional distributional discourse analysis. In M. Lewis, B. Coecke, J. Hedges, D. Kartsaklis, and D. Marsden, editors, Procs. of the 2018 Workshop on *Compositional Approaches in Physics, NLP, and Social Sciences*, volume 283 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–12, 2018.

[19] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi. Foundations for near-term quantum natural language processing, 2020.

[20] B. Coecke and A. Kissinger. *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning.* Cambridge University Press, 2017.

[21] B. Coecke, M. Lewis, and D. Marsden. Internal wiring of cartesian verbs and prepositions. In M. Lewis, B. Coecke, J. Hedges, D. Kartsaklis, and D. Marsden, editors, Procs. of the 2018 Workshop on *Compositional Approaches in Physics, NLP, and Social Sciences*, volume 283 of *Electronic Proceedings in Theoretical Computer Science*, pages 75–88, 2018.

[22] B. Coecke and K. Meichanetzidis. Meaning updating of density matrices. *arXiv:2001.00862*, 2020.

[23] B. Coecke and É. O. Paquette. Categories for the practicing physicist. In B. Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics, pages 167–271. Springer, 2011. arXiv:0905.3010.

[24] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning. In J. van Benthem, M. Moortgat, and W. Buszkowski, editors, *A Festschrift for Jim Lambek*, volume 36 of *Linguistic Analysis*, pages 345–384. 2010. arxiv:1003.4394.

[25] B. Coecke and V. Wang. Grammar equations. *Accepted for SEMSPACE*, 2021.

[26] T. Duneau. Solving logical puzzles in DisCoCirc, 2020.

[27] J. Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[28] E. Grefenstette and M. Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *The 2014 Conference on Empirical Methods on Natural Language Processing.*, pages 1394–1404, 2011. arXiv:1106.4058.

[29] V.N. Grishin. On a generalization of the Ajdukiewicz-Lambek system. In *Studies in nonclassical logics and formal systems*, pages 315–334. Nauka, Moscow, 1983.

[30] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and philosophy*, 14(1):39–100, 1991.

[31] H. Kamp and U. Reyle. *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, volume 42. Springer Science & Business Media, 2013.

[32] D. Kartsaklis and M. Sadrzadeh. Prior disambiguation of word tensors for constructing sentence vectors. In *The 2013 Conference on Empirical Methods on Natural Language Processing.*, pages 1590–1601. ACL, 2013.

[33] D. Kartsaklis and M. Sadrzadeh. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*. Kyoto Japan, 2014.

[34] D. Kartsaklis, M. Sadrzadeh, S. Pulman, and B. Coecke. Reasoning about meaning in natural language with compact closed categories and Frobenius algebras. In *Logic and Algebraic Structures in Quantum Computing and Information*. Cambridge University Press, 2015. arXiv:1401.5980.

[35] T. Kuhn. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1):121–170, March 2014.

[36] J. Lambek. The mathematics of sentence structure. *American Mathematics Monthly*, 65, 1958.

[37] J. Lambek. Type grammar revisited. *Logical Aspects of Computational Linguistics*, 1582, 1999.

[38] J. Lambek. From word to sentence. *Polimetrica, Milan*, 2008.

[39] Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.

[40] M. Lewis. Compositionality for recursive neural networks. arXiv:1901.10723, 2019.

[41] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsalkis, and B. Coecke. Qnlp in practice: Running compositional models of meaning on a quantum computer. *arXiv preprint arXiv:2102.12846*, 2021.

[42] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

[43] K. Meichanetzidis, A. Toumi, G. de Felice, and B. Coecke. Grammar-aware question-answering on quantum computers. *arXiv preprint arXiv:2012.03756*, 2020.

[44] F. Meyer and M. Lewis. Modelling lexical ambiguity with density matrices. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 276–290, 2020. arXiv:2010.05670 [cs.CL].

[45] R. Montague. Universal grammar. *Theoria*, 36:373–398, 1970.

[46] R. Montague. The proper treatment of quantification in ordinary English. In *Approaches to natural language*, pages 221–242. Springer, 1973.

[47] R. Muskens. Categorial grammar and discourse representation theory. In *15th International Conference on Computational Linguistics, COLING 1994, Kyoto, Japan, August 5-9, 1994*, pages 508–514, 1994.

[48] T. Polajnar, L. Rimell, and S. Clark. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11, 2015.

[49] A. Preller. Category theoretical semantics for pregroup grammars. In *International Conference on Logical Aspects of Computational Linguistics*, pages 238–254. Springer, 2005.

[50] A. Radford. *Transformational Grammar: A First Course*. Cambridge University Press, May 1988.

[51] M. Sadrzadeh, S. Clark, and B. Coecke. The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Journal of Logic and Computation*, 23:1293–1317, 2013. arXiv:1404.5278.

[52] M. Sadrzadeh, S. Clark, and B. Coecke. The Frobenius anatomy of word meanings II: possessive relative pronouns. *Journal of Logic and Computation*, 26:785–815, 2016. arXiv:1406.4690.

[53] M. Sadrzadeh and R. Muskens. Static and dynamic vector semantics for lambda calculus models of natural language. *arXiv:1810.11351*, 2018.

[54] P. Selinger. A survey of graphical languages for monoidal categories. In B. Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics, pages 275–337. Springer-Verlag, 2011. arXiv:0908.3347.

[55] M. Steedman. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5:403–439, 1987.

[56] L. Tesniere. *Elements of Structural Syntax*. John Benjamins Publishing Company. Publication Title: z.185.

[57] A. Toumi. Categorical compositional distributional questions, answers & discourse analysis. Master's thesis, University of Oxford, 2018.

[58] A. Visser. Contexts in dynamic predicate logic. *Journal of Logic, Language and Information*, 7(1):21–52, 1998.

[59] G. Wijnholds and M. Sadrzadeh. Classical copying versus quantum entanglement in natural language: The case of vp-ellipsis. *arXiv:1811.03276*, 2018.

[60] G. Wijnholds and M. Sadrzadeh. Evaluating composition models for verb phrase elliptical sentence embeddings. In *Proceedings of the 2019 Conference of the ACL*, pages 261–271. Association for Computational Linguistics, 2019.

[61] G. Wijnholds, M. Sadrzadeh, and S. Clark. Representation learning for type-driven composition. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 313–324, 2020.

[62] A. Wyner, K. Angelov, G. Barzdins, D. Damljanovic, B. Davis, N. Fuchs, S. Hoefler, K. Jones, K. Kaljurand, T. Kuhn, M. Luts, J. Pool, M. Rosner, R. Schwitter, and J. Sowa. On Controlled Natural Languages: Properties and Prospects. pages 281–289, June 2010.

[63] R. Yeung and D. Kartsaklis. A CCG-based version of the DisCoCat framework. *Accepted for SEMSPACE*, 2021.