Parsing Conjunctions in DisCoCirc

Tiffany Duneau
University of Oxford
fanny.duneau@cs.ox.ac.uk

Abstract

In distributional compositional models of meaning logical words require special interpretations, that specify the way in which other words in the sentence interact with each other. So far within the DisCoCat framework, conjunctions have been implemented as merging both conjuncts into a single output, however in the new framework of DisCoCirc merging between nouns is no longer possible. We provide an account of conjunction and an interpretation for the word *and* that solves this, and moreover ensures certain intuitively similar sentences can be given the same interpretations.

1 Introduction

The distributional semantics paradigm allows us to model the meanings of words in terms of their surrounding words, as well as the meaning of texts in terms of their constituent words. Meanwhile compositional semantics considers how meaning can arise from the grammatical form of a sentence. The distributional compositional (DisCo-) approach to modelling meaning introduced by Coecke et al. (2010) allows the meaning of a sentence to be computed as a function of both the distributional meaning of the words involved, as well as its grammatical form. Recent work by Lorenz et al. (2021) has implemented the approach on quantum hardware, and Kartsaklis and Sadrzadeh (2016); Lewis (2019) have shown the classical model to perform well at various natural language processing tasks.

However, there are certain words whose meaning cannot be expressed as a function of their surrounding words - logical words like *and, or* and *not*, as well as pronouns such as *whom* and *that* - since they tend to appear in all contexts and thus render distributional approaches to modelling them inadequate. On the other hand, such words typically have an impact on the way in which the other words in the sentence interact, and thus take on a more syntactic role, acting as an extension of the grammar. In this paper, we shall focus in particular on the logical connective *and*.

Previously, in the DisCo- formalisms conjunctions have been interpreted as simply mixing the conjuncts involved together to produce a single entity of the appropriate type: as described in Kartsaklis (2016), we can mix the adjectives *red* and *yellow* together to make a new adjective that describes things that are both red and yellow. This approach works less well however, when we attempt to mix nouns; when we discuss *a hat and a scarf*, we are not discussing a single hybrid object that is both a hat and a scarf, but *two* objects, one of which is a hat, the other of which is a scarf. This difference suggests that there are two different types of conjunctions at play. The DisCoCirc framework allows us to express this difference - in the introductory paper Coecke (2020), they are denoted as *linear* and *non-linear* forms of *and*.

The non-linearity can be seen as arising from an induced duplication:

1a Alice wears red and yellow socks.

b Alice wears red socks. Alice wears yellow socks.

2a Alice wears a hat and a scarf.

b Alice wears a hat. Alice wears a scarf.

Sentence (1a) employs a linear form of *and* - in this case both adjectives are applied to the socks which then feeds into the rest of the sentence. We can contrast this with the non-linear interpretation of the same sentence in (1b), which does not seem to convey the same meaning. On the other hand, (2a) seems to carry the same meaning as (2b) which exhibits the implicit duplication of *wear* induced by the conjunction. This distinction was previously lost as conjoined nouns were considered equivalent to a single noun - this meant that any duplication would have occurred at best implicitly.

Standard DisCo- approaches are often limited to linear operations only, as they have standardly been implemented using vector spaces and linear maps, however certain phenomena seem to require non-linear elements: in Lewis (2020), negation is modelled as a (non-linear) projection of positive maps onto their orthogonal subspaces, while (Wijnholds and Sadrzadeh, 2019a) shows how modelling duplication *explicitly* improves performance in sentence similarity and entailment tasks involving ellipsis and anaphora. Wijnholds and Sadrzadeh (2019b) describes how the duplication is introduced, by augmenting the basic pregroup based grammars with a non-linear reduction rule.

Here, we will also be introducing a non-linear element, modifying word meanings. In particular, this will allow us to model the duplication in sentences like (1) and (2), such that the interpretation of sentences we consider equivalent are the same when expressed using the DisCoCirc framework. We will first give a brief overview of this framework, then introduce the required structures and definitions for modelling *and*, finishing with some examples.

2 Mathematical Background

We will give a brief overview of the mathematical background for the DisCo- approaches to modelling meaning. For a detailed introduction to the category theory see Heunen (2020), while Coecke and Kissinger (2017); Selinger (2009) provide a more diagrammatic treatment. Coecke et al. (2010); Coecke (2020) introduce the DisCoCat and DisCoCirc frameworks respectively.

2.1 Compact Closed Categories

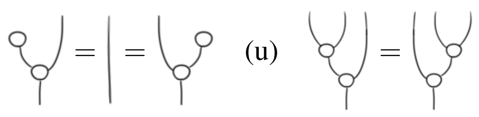
We will be encoding sentences as diagrams, in which wires will carry meanings, and boxes will represent processes that modify these meanings. These diagrams are representations of structures in a compact closed category; we will take the convention of reading the diagrams from top to bottom, and will sometimes add a direction to the wire to indicate the presence of a dual. The diagrams come with an associated calculus, framed as a set of rewrite rules. Diagrams and the structures they represent are equivalent if and only if there is a valid way to rewrite one into the other.

We can use such diagrams to express syntactic relations between the words of a sentence. In the present case, this is what we are most interested in, as conjunctions impart meaning mostly by imposing extra syntactic dependencies between other words in the sentence. In order to get back a representation of what the diagram *means*, however, we also need to supply a way to *interpret* diagrams, often within a specific compact closed category. Standardly, this has been as linear maps between vector spaces or relations between sets, though other categories have been used too (Bankova et al., 2016; Bolt et al., 2017). Here, we will not be concerned with the specific meanings, only the ways in which the words are connected. As such, we can take ourselves to be working in the category freely generated by a chosen set of types and boxes, along with the required cups, caps and swaps that make the category compact closed.

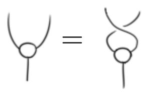
2.2 Monoids

Monoids encapsulate the idea of combining and splitting objects, and so form a vital part of our theory of meaning, and are particularly relevant to the notion of conjunction.

Definition 1. *A* ***monoid*** $(A, \curlyvee, \circ)$*, is structure over an object* A *that satisfies unitality (u) and associativity (a) axioms:*


(u) (a)

Commutative monoids satisfy the additional condition (c):


(c)

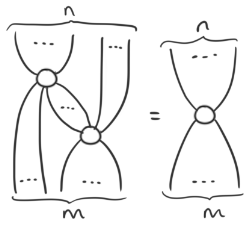
A **comonoid** structure $(A, \curlywedge, \bullet)$, is a vertically flipped version of the monoid, satisfying the analogously flipped axioms.

2.2.1 Spiders

Certain monoid-comonoid pairs satisfy additional rules[1], which allows us to write the dots all in the same colour, as they may be interchanged. In particular, as associativity identifies all orders of composition, and we may write sequences of monoid

[1]Namely, when the comonoid is the dagger of the monoid and they are special and Frobenius: Heunen (2020) chapter 5

or comonoid applications as a single dot with many legs. These dots obey the 'spider fusion' rule:



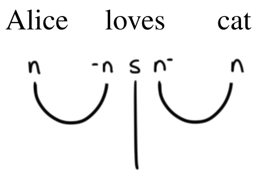
(SF)

2.3 Pregroups

Having created all the above structure to encode meaning, we now turn to grammar. A *pregroup* is a group where we may have distinct left and right inverses for each element. Extending this into pregroup grammar in the style of Lambek (1968) and Bar-Hillel (1953), we start with some generating elements, or basic grammatical types: 's' (sentence) and 'n' (noun), along with a unit element I, and their respective left and right inverses. More complex types are composed by stringing the basic types together via the multiplication operation: transitive verbs such as *loves* and *sits on* are of the form '$^{-}n\ s\ n^{-}$', as they need a noun on either side to form a sentence. The multiplication of the pregroup is given by a partial order on the strings generated by the basic types:

$$x \cdot {}^{-}x \leq I \qquad x^{-} \cdot x \leq I$$
$$I \leq {}^{-}x \cdot x \qquad 1 \leq x \cdot x^{-}$$

These equations correspond exactly to the cups and caps within a compact closed category if we take I to be the monoidal unit. Indeed the category formed by taking the grammatical types as objects, string concatenation as parallel composition, and the reductions to define morphisms between objects is compact closed. We can hence write grammatical reductions out graphically. '*Alice loves [the] cat*' has grammatical type '$n\ {}^{-}nsn^{-}\ n$', and reduces as follows:



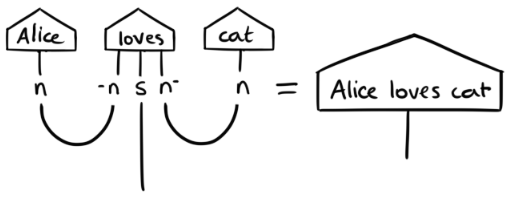
If, as with this example there is a way to reduce the grammatical type to a single sentence wire, then the sentence is grammatical. Of note is that there may sometimes be different ways in which the same string can be reduced. These correspond to different possible grammatical interpretations of the sentence. In forming reductions, we are not allowed to cross wires over each other, as this would allow some non grammatical strings such as '*loves cat Alice*' to be reduced too, by swapping word order as part of the reductions.

2.4 DisCoCat

DisCoCat is a *dis*tributional *co*mpositional *cat*egorical model of meaning, that is formulated in a compact closed category, introduced in Coecke et al. (2010). The distributional aspect of the model concerns word meaning, often encoded as vectors. The easiest way of combining our words into a sentence is to simply write them next to each other:

Alice
loves
cat

However this fails to capture any meaningful grammatical relationship between the words, other than perhaps word order. We hence add a grammatical type to each of our wires, and allow the grammar to mediate how the words *compose* to give the final sentence meaning, linking the wires together using cups and caps:

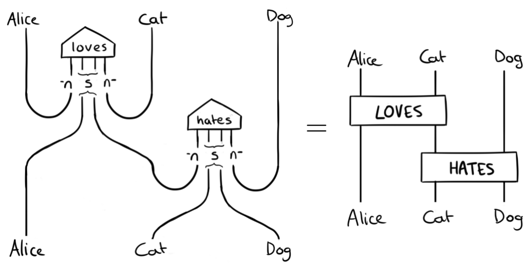


As mentionned above, there are some words for which we hard code the meaning - logical words like *and* and *not* (Kartsaklis, 2016), as well as relative pronouns like *which* and *whom* (Sadrzadeh et al., 2013, 2016). In many of these cases the 'copy' spider has been used, which copies or merges wires relative to a given basis. Importantly however, this spider is a *linear* operation so cannot truly duplicate anything.

Some words will also need to be assigned an ambiguous grammatical type, if they can occur in different contexts; the correct type to use can be informed by considering the choice that allows the surrounding sentence to be reduced into a sentence.

2.5 DisCoCirc

DisCoCat allows us to encode single sentences to obtain a meaning vector describing the entire sentence. Going one step further, DisCoCirc allows us to compose multiple sentences together, and model the meaning of an entire text. This involves shifting our perspective slightly, so that we are considering the way the meaning of words are altered by a sentence, for example the way a character's name changes meaning as we read a novel and learn more about them. Rather than having a single sentence type, we hence move to taking our types to be the *dynamic objects*, like *The cat* or *Alice*, that feature in the sentences we are concerned with. Each sentence is then still tied together in much the same way as in DisCoCat - according to the grammar - but the outgoing sentence type is now a composite of the dynamic objects. The overall sentence becomes a box that acts on these dynamic objects, rather than just a state, allowing it to be composed with other sentences. A sentence acts as the identity on any object that is not involved in it. For example, we can now encode '*Alice loves the cat. The cat hates the dog.*' as follows:



Since we are now composing sentences together, we allow wires to cross freely *between* sentences, though no extra crossings may occur in the grammatical reductions.

By encoding the meaning of texts in terms of the meanings of the characters within them, we sacrifice having a common space in which all texts are encoded, and all the associated benefits for sentence comparison. This remains an interesting move to make if we are interested in making a more involved analysis of the *content* of the texts in question, as we can retain a rather detailed representation of what is happening without needing to squeeze everything down into a fixed space. This has a particular impact for the modelling of and, as it will require us to introduce a notion of duplication.

3 New structure

In addition to the basic structures presented above, we shall make use of a few extra components in order to model and.

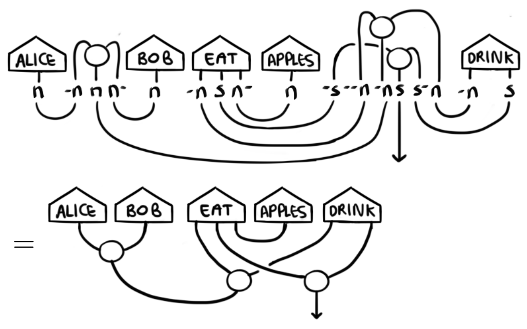
3.1 Fine-graining of the n type

Much of the work done with DisCoCat and DisCoCirc so far makes use of only n and s as generators for the grammar. In DisCoCirc, it becomes necessary to break the usual n type down into two different types, so that we can treat them differently when encoding them. The distinction we need is between *singular* (which we will keep as n) and *plural* (denoted N) nouns[2]. Singular nouns behave just as we expect, since they contain exactly one noun wire, whereas plural nouns are in fact a series of singular nouns that have been packaged together. Formally, this 'packaging together' occurs via the monoidal tensor, and indeed the sentence type in DisCoCirc corresponds to the plural noun type N. We will still make use of the sentence type s, to ensure that the grammar is not resolved differently, and so that no grammatically incorrect sentences are accepted as a result.

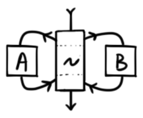
3.2 Merge box

In order to model and, we will need a notion of merging. Previously (Kartsaklis, 2016), a spider was used:

- Alice and Bob eat apples and drink.



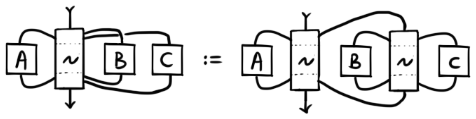
However, the notion of merging a pair of boxes is much more general than this particular choice. Indeed, a series of methods for combining density matrices has been surveyed in Cuevas et al. (2020); Coecke and Meichanetzidis (2020). The basic form of such a map is as follows:



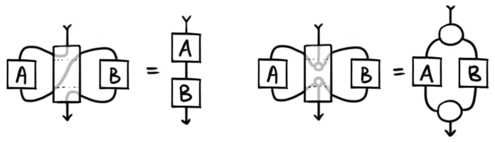
Here we have taken some liberties in drawing the diagram connections - concretely the wires coming

[2]In Lambek (1968), a different sort of plural noun is considered, typed n^*, which seems more aimed at dealing with generic nouns (*men*; *strawberries*) rather than multiple copies of n (*two strawberries*; *men and women*) as introduced here.

out of the sides of the box should be considered as coming out of the top or bottom (whichever is closest), however we draw them differently to highlight them as 'intermediate' wires, each of which is representing a particular branch that will be merged. Merging more than two wires together is defined recursively in terms of a two way merge:

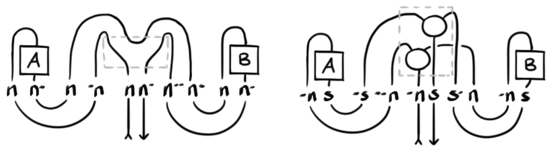


In order to obtain a useful merging operation, we will restrict our attention to associative binary operations - semigroups - and in particular consider two natural examples (where SPIDER corresponds to Mult in Cuevas et al. (2020), and SEQ is sequential composition.):

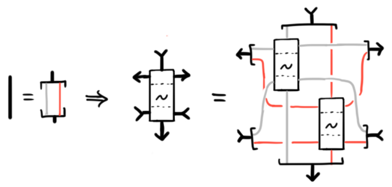


(SEQ) (SPIDER)

On top of being associative, these operations are actually based on monoids - the copy spider and 'pair of pants' monoid respectively. This form is most clear when we draw the merge with it's wires paired in a particular way:



We extend the merging to plural wires by merging each component wire separately[3]:



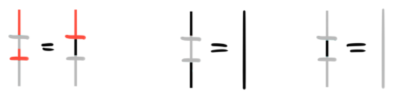
3.3 Duplication 2-Morphism

The main ingredients that will help us formulate conjunctions are a 2-morphism, or meta operation which we will denote $[//]$, along with a marker morphism $//$, and colour typing morphisms $\vdash, \dashv$.

[3]The N typed wires have been drawn thicker to highlight that they are actually a collection of wires. The bracketing or gathering morphism should only be taken as a notational tool that highlights which specific n wires contribute to the plural N wire.

3.3.1 Wire colours

In order to control the 'footprint' of the $[//]$ morphism, we make use of certain extra typing annotations, which we will represent using wire colours. These annotations stack rather than mix - a red and blue wire is not the same as a purple wire. We take 'uncoloured' wires to be black, with the colours controlled by 'start' ($\vdash$), and 'stop' ($\dashv$) morphisms.

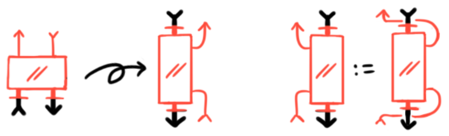


Commute Identity

As these colours have nothing to do with the meanings carried by the wire, we can view them as living in a separate sub-wire, conjoined to the main meaning wire. Supposing that the compact closed category $\boldsymbol{Col}$ admits colour morphisms like the above, and that our main 'semantic' category is $\boldsymbol{S}$, we can define a new category of semantics with colour annotations: $\boldsymbol{S} \times \boldsymbol{Col}$. This new category will also be compact closed, inheriting the relevant structure from the compact closure of both $\boldsymbol{S}$ and $\boldsymbol{Col}$, and can hence replace $\boldsymbol{S}$ as the category in which we are expressing our meanings.

3.3.2 Marker morphism

Each time we want to duplicate a wire, we will introduce a 'marker' morphism, $//$, to indicate where the duplication should occur. Such a morphism will have an associated colour, and type signature to indicate how the wires are to be split by the duplication operation. We will draw it as follows[4]:



The thick N or S type wires contain the series of n wires to duplicate over, while the thin coloured wire highlights the part of the diagram in need of duplication, standing in for a given n wire.

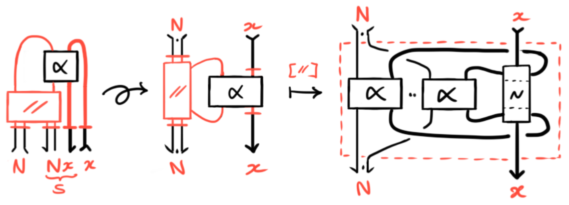
3.3.3 The duplication 2-morphism

Informally, a **2-morphism** is a morphism defined between the *morphisms* (rather than objects) of a category. Analogously, it can be viewed as specifying an extra rewrite rule for our diagrams - the new diagram obtained may not be strictly equal to the

[4]When expressing word states using $//$ we will tend to write it horizontally, whereas within a full DisCoCirc diagram it will be more convenient to write the morphism vertically.

previous one, as per the usual rewrite rules, however the 2-morphism defines a sense in which the diagrams should be viewed as equivalent. In our case, we will be introducing a 2-morphism that will deal with duplication, as there is no corresponding (1-)morphism in a compact closed category that can do the job.

Definition 2. *The 2-morphism [//] copies boxes and introduces a mixing operation as follows:*



Importantly, we apply $[//]$ centered on a marker, exclusively to the largest sub-diagram that has wires coloured the same way as the marker. For this to be well defined, we hence require that each instance of $//$ be uniquely coloured. The resulting diagram applies the coloured sub-diagram to each marked wire, merging any other wires together according to a chosen merge operation.

If the underlying mixing operation is both associative and commutative, the order in which we apply the $[//]$ is associative also (for a proof see the appendix). In such a case, there will be a unique normal form for the diagram that contains no marker boxes.

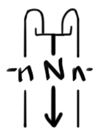
A full expansion of the marker boxes in a diagram renders any leftover colour annotations irrelevant to the meaning of the sentence. The final step in computing the DisCoCirc diagram corresponding to a given sentence involving such markers will then be to apply a forgetful functor from $\boldsymbol{S} \times \boldsymbol{Col}$ into $\boldsymbol{S}$, removing the now unnecessary colour subwire.

4 Modelling *and*

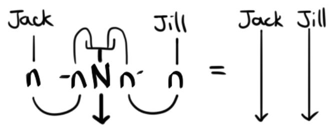
Having set the scene, we are now equipped to start modelling the word and. The first aspect to note, is that and has a variable type, of the form ${}^{-}x\ x\ x^{-}$, where x can stand for any (possibly composite) grammatical type. The account we give will hence be equally generic, though we proceed mostly via example.

4.1 Looking inside the box

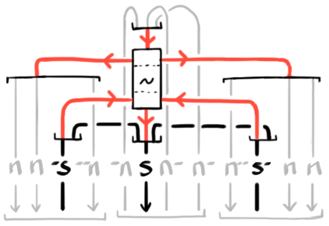
In the most basic case, conjoining two nouns is just a case of packaging them next to each other:



- Jack and Jill.

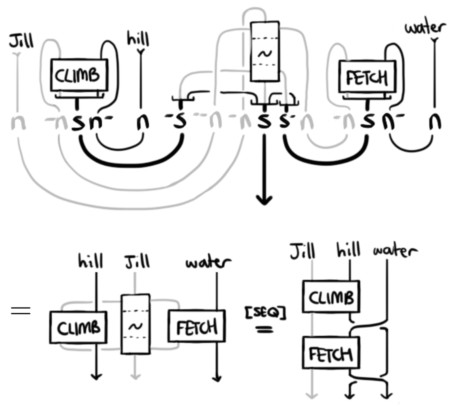


Next, conjoining boxes is just a case of merging them together. In some cases, the components we want to merge may not involve the same nouns - in order for the types to match we cannot merge them, so must combine them into the outgoing sentence wire directly. These extra nouns are drawn in black below:

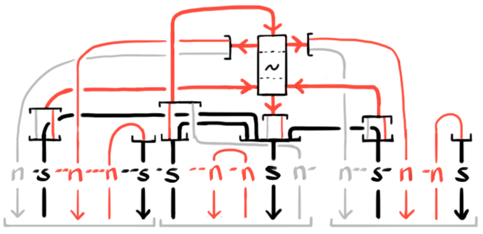


By varying the number of noun wires coming in or out appropriately, we can use the above to construct forms for adjectives ($n\ n^{-}$), simple verbs (${}^{-}n\ s$), transitive verbs (${}^{-}n\ s\ n^{-}$), and di-transitive verbs (${}^{-}n\ s\ n^{-}\ \ n^{-}$). Removing all the nouns results in a sentence combiner that acts analogously to the simple noun case. To illustrate, consider:

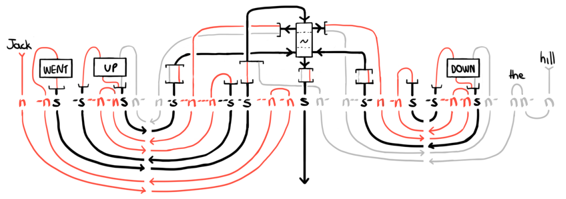
- Jill climbs hills and fetches water.

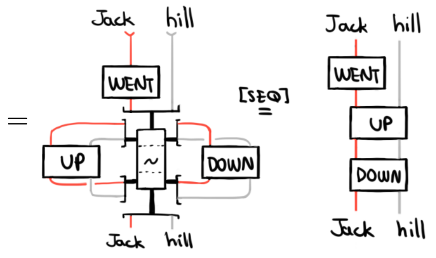


Another important structure is prepositions, typed $^{-}s\ ^{--}n\ ^{-}n\ s\ n^{-}$. In this case, a specific subject noun (in red), and an indirect object (in light grey) are identified in advance, and are subsequently passed to the two conjuncts to be mixed. Again, we collect any extra nouns picked up along the way and ensure they bypass the merging. Similarly to above, we can also add or remove as many light grey nouns as necessary to type match, following the schema given.



- Jack went up and down the hill.





The internal wirings for other types will largely follow the same format, merging together wires that are involved on both sides of the conjunct and combining the rest into an outgoing N or s plural noun.

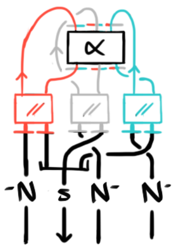
4.2 Dealing with N

The way in which we are conjoining nouns is introducing a new plural noun type N. In order for our sentence parsings to keep working, we hence need to introduce new forms for our other boxes that include N where we would usually have n. Bearing in mind that our goal is for the grammar wirings to look the same, this means that we are looking to convert N into n internally to the box in question.

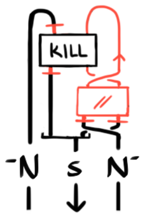
The intuition behind this split is that usually, the action described by a box is not shared between the parties involved - instead each character does the given action independently. For example, consider the following sentences:

1. Alice and Bob eat cucumber.
2. Alice eats cucumber. Bob eats cucumber.
3. Alice and Bob murder Charlie.
4. Alice murders Charlie. Bob murders Charlie.

In (1), Alice and Bob are not sharing the same '*eating*', and the sentence seems equivalent to (2), suggesting that the verb is simply duplicated to accommodate the multiple subjects. On the other hand, in (3), there is only one murder, so Alice and Bob must be doing it together, suggesting that in this case we cannot duplicate the single-subject form. Indeed, (4) seems somewhat contradictory - when we learn that Alice murders Charlie, we assume that Charlie is then dead. Subsequently learning that Bob murders Charlie too appears odd, since as far as we are concerned there is no Charlie left for Bob to kill. We can express this difference with the following expansions of the verb boxes:



Generic verb

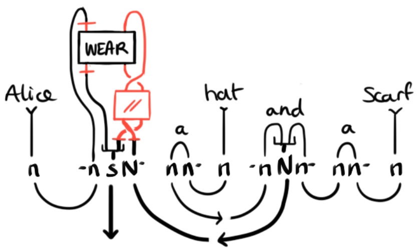


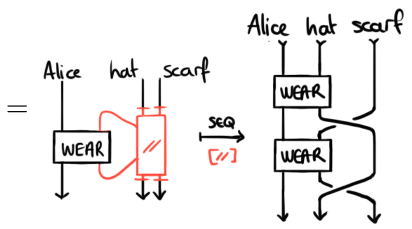
Kill

4.3 Examples

Having established the theory, we can now illustrate how this works with some actual sentences. First, consider the non-linear example given at the start:

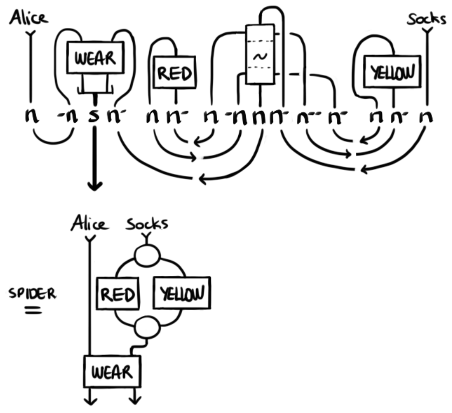
- Alice wears a hat and a scarf.





In this case, the hat and scarf are first packaged into a plural noun, which when fed into the verb *wears* induces a duplication. Taking the merge operation to be given by (SEQ), the final diagram obtained is equivalent to that of '*Alice wears a hat. Alice wears a scarf*'.

- Alice wears red and yellow socks.



In this linear case, the key operation is the merge - no duplication is necessary. The natural choice in this case seems to be (SPIDER), which puts both sides in parallel, suggesting that perhaps (SPIDER) should be used within *and* boxes, whilst (SEQ) should be the merge introduced by the duplication.

In taking the non-commutative (SEQ) as our merge operation, however, we will sometimes need to make an arbitrary choice about the order in which we apply [//]. This problem notably occurs with di-transitive verbs:

- Alice and Bob show Mary apples and pears.

In such cases, the various expansions are arguably different, as they seem to introduce a temporal ordering on the events described, which is not present when there is no common wire connecting the duplicates:

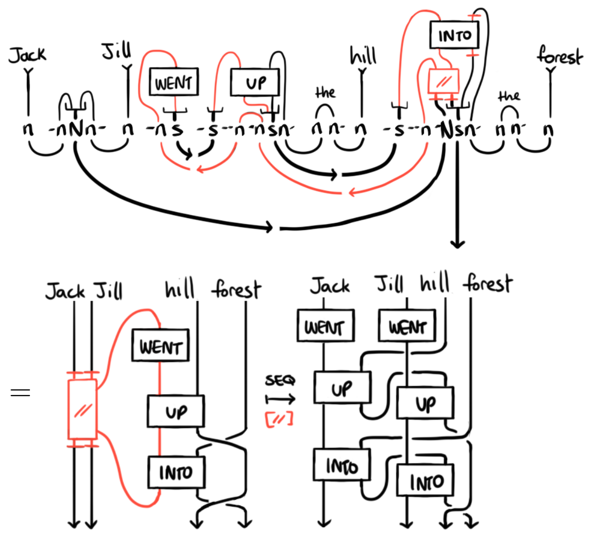
- Alice shows Mary apples. Alice shows Mary pears. Bob shows Mary apples. Bob shows Mary pears.

- Alice shows Mary apples. Bob shows Mary apples. Alice shows Mary pears. Bob shows Mary pears.

The solution would then either be to use a commutative merge like (SPIDER), which would result three different sentences; or to keep (SEQ), but attribute the non-commutativity to the verb *show* instead. In this case, it seems we should model *show* like *kill*, suggesting a link between whether an expansion of the verb is induced by *and*, and whether the verb's meaning is closely linked to the relative time at which it occurs.

Next, we consider a slightly more involved sentence which exhibits the way in which the plural nouns are fed through the sentence:

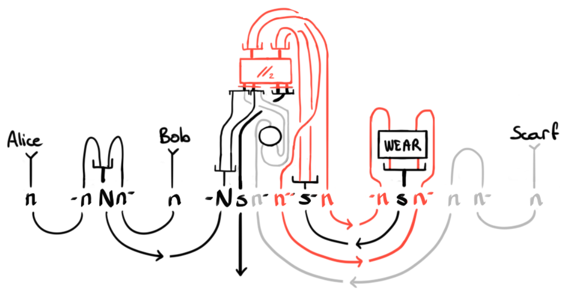
- Jack and Jill went up the hill into the forest.

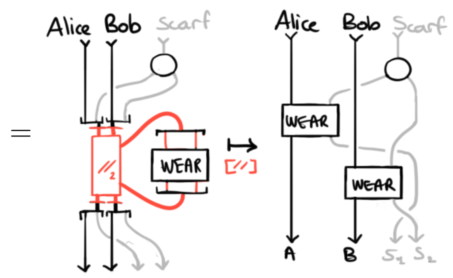


In this case, we can see that the plural noun is fed into the sub-phrase in which it occurs via the marker morphism, so that the sentence expands into the same diagram that would be given by '*Jack went up the hill into the forest. Jill went up the hill into the forest*', as expected.

The duplication operation can also be used with paired wires. In this case, the pairs will be fed through the inner phrase together, rather than independently. For example, consider:

- Alice and Bob *each* wear a scarf.

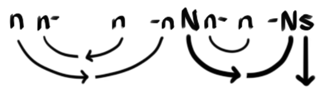




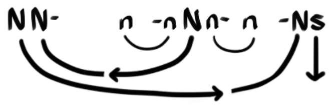
In this case, we have provided a particular interpretation of *each* that coordinates the subject nouns with a copy of the object noun before being fed through the marker morphism. This ensures that Alice and Bob are wearing different copies of the scarf, rather than both wearing the same scarf, or both wearing both scarves. Taking advantage of this distinction also seems like a promising way to model words with similar effects, such as *respectively*.

Finally, in the framework presented we can also account for conjunctive ambiguity, which arises as different ways of bracketing the conjuncts:

(Clumsy Jack) and Jill fell.



Clumsy (Jack and Jill) fell.



The ambiguity here is resolved by our choice of where to place the plural N types. In the first case, we assign *clumsy* a singular adjective type ($n\ n^{-}$), whereas in the second sentence it is plural ($N\ N^{-}$). Making sure to index the N types with the number of wires they are encapsulating, the grammatical ambiguity present can hence be isolated to the type assignment rather than the parsing tree.

5 Related Work

The non-linearity introduced here has very similar effects to the non-linearity Wijnholds and Sadrzadeh (2019b) introduce to deal with ellipsis, however arises rather differently. In particular, they introduce the non-linearity to the grammar parsing, effectively by allowing parts of the derivation to freely be reused elsewhere. In this way, arbitrary parts of the sentence can get copied over to a new location. In contrast, the duplication introduced here is via the words involved, such that the duplication can only occur along the wires given by the (linear) pregroup grammar. The duplication also does not apply to wires themselves - only boxes get copied, making this a strictly weaker notion.

Though we have not discussed a particular choice of meaning embedding, the approach relates to work encoding logical words using vector-based semantics. Within DisCoCirc, we are primarily concerned with *characters* - treating the nouns involved as individual entities. In some cases, however we might be interested in nouns as *concepts*, for which conjunction does not seem to imply the presence of multiple characters, but instead suggests a merging of the component concepts. Aerts (2009) discusses such concepts, in particular considering the so-called *Guppy effect* (*Guppy* is considered a good example of the concept *pet fish* despite being a bad example of both *fish* and *pet* when the concepts are considered separately). He suggests representing the joint concept *pet and fish* as a superposition in a Fock space to account for this: $|\textit{pet and fish}\rangle := (|\textit{pet}\rangle \otimes |\textit{fish}\rangle) \oplus (|\textit{pet}\rangle + |\textit{fish}\rangle)$ In the account of conjunction given here, we effectively split the two ways of combining concepts: the tensor is used on ‘characters’, while everything else is merged - potentially as a superposition.

6 Conclusion

In summary, we have provided an account of *and* within the framework of DisCoCirc, that allows us to generate diagrams capturing the intuitive meaning of sentences that involve conjunctions. Due to the non-linear nature of *and*, we associate it to a diagram rewriting operation or 2-morphism, that duplicates the relevant parts of our diagrams.

The treatment provided allows us to parse the common usage of *and*, however there is more work to be done when it comes to certain more complex cases. In particular, there are many related words that control how and whether to duplicate words, such as *respectively* and *each*, as well as certain phrases like *three times* which interact with duplication in more complex ways. Ellipsis is also a closely related grammatical notion, and it would be interesting to see if the duplication approach explored here can provide a suitable solution.

References

Diederik Aerts. 2009. Quantum Structure in Cognition. *Journal of Mathematical Psychology*, 53(5):314–348. ArXiv: 0805.3850.

Desislava Bankova, Bob Coecke, Martha Lewis, and Daniel Marsden. 2016. Graded Entailment for Compositional Distributional Semantics. *arXiv:1601.04908 [quant-ph]*. ArXiv: 1601.04908.

Yehoshua Bar-Hillel. 1953. A Quasi-Arithmetical Notation for Syntactic Description. *Language*, 29(1):47.

Joe Bolt, Bob Coecke, Fabrizio Genovese, Martha Lewis, Dan Marsden, and Robin Piedeleu. 2017. Interacting Conceptual Spaces I : Grammatical Composition of Concepts. *arXiv:1703.08314 [cs]*. ArXiv: 1703.08314.

Bob Coecke. 2020. The Mathematics of Text Structure. *arXiv:1904.03478 [quant-ph]*. ArXiv: 1904.03478.

Bob Coecke and Aleks Kissinger. 2017. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, Cambridge.

Bob Coecke and Konstantinos Meichanetzidis. 2020. Meaning updating of density matrices. *arXiv:2001.00862 [quant-ph]*. ArXiv: 2001.00862.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical Foundations for a Compositional Distributional Model of Meaning. *arXiv:1003.4394 [cs, math]*. ArXiv: 1003.4394.

Gemma De las Cuevas, Andreas Klingler, Martha Lewis, and Tim Netzer. 2020. Cats climb entails mammals move: preserving hyponymy in compositional distributional semantics. *arXiv:2005.14134 [cs, math]*. ArXiv: 2005.14134.

Christiaan Johan Marie Heunen. 2020. *Categories for quantum theory: an introduction [electronic resource]*, first edition. edition. Oxford graduate texts in mathematics. University Press, Oxford.

Dimitri Kartsaklis. 2016. Coordination in Categorical Compositional Distributional Semantics. *Electronic Proceedings in Theoretical Computer Science*, 221:29–38. ArXiv: 1606.01515.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2016. Distributional Inclusion Hypothesis for Tensor-based Composition. *arXiv:1610.04416 [cs]*. ArXiv: 1610.04416.

Joachim Lambek. 1968. The Mathematics of Sentence Structure. *Journal of Symbolic Logic*, 33(4):627–628.

Martha Lewis. 2019. Modelling hyponymy for DisCoCat.

Martha Lewis. 2020. Towards logical negation for compositional distributional semantics. *arXiv:2005.04929 [cs, math]*. ArXiv: 2005.04929.

Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis, and Bob Coecke. 2021. QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer. *arXiv:2102.12846 [quant-ph]*. ArXiv: 2102.12846.

Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. 2013. The Frobenius anatomy of word meanings I: subject and object relative pronouns. *Journal of Logic and Computation*, 23(6):1293–1317. ArXiv: 1404.5278.

Mehrnoosh Sadrzadeh, Stephen Clark, and Bob Coecke. 2016. The Frobenius anatomy of word meanings II: possessive relative pronouns. *Journal of Logic and Computation*, 26(2):785–815. ArXiv: 1406.4690.

Peter Selinger. 2009. A survey of graphical languages for monoidal categories. *arXiv:0908.3347 [math]*. ArXiv: 0908.3347.

Gijs Wijnholds and Mehrnoosh Sadrzadeh. 2019a. Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 261–271, Minneapolis, Minnesota. Association for Computational Linguistics.

Gijs Wijnholds and Mehrnoosh Sadrzadeh. 2019b. A Type-Driven Vector Semantics for Ellipsis with Anaphora Using Lambek Calculus with Limited Contraction. *Journal of Logic, Language and Information*, 28(2):331–358.