A CCG-Based Version of the DisCoCat Framework

Richie Yeung and **Dimitri Kartsaklis**

Cambridge Quantum Computing Ltd.
17 Beaumont Street, Oxford, OX1 2NA, UK

{richie.yeung;dimitri.kartsaklis}@cambridgequantum.com

Abstract

While the DisCoCat model (Coecke et al., 2010) has been proved a valuable tool for studying compositional aspects of language at the level of semantics, its strong dependency on pregroup grammars poses important restrictions: first, it prevents large-scale experimentation due to the absence of a pregroup parser; and second, it limits the expressibility of the model to context-free grammars. In this paper we solve these problems by reformulating DisCoCat as a passage from Combinatory Categorial Grammar (CCG) to a category of semantics. We start by showing that standard categorial grammars can be expressed as a bi-closed category, where all rules emerge as currying/uncurrying the identity; we then proceed to model permutation-inducing rules by exploiting the symmetry of the compact closed category encoding the word meaning. We provide a proof of concept for our method, converting "Alice in Wonderland" into DisCoCat form, a corpus that we make available to the community.

1 Introduction

The compositional model of meaning by Coecke, Sadrzadeh and Clark (Coecke et al., 2010) (from now on DisCoCat[1]) provides a conceptual way of modelling the interactions between the words in a sentence at the level of semantics. At the core of the model lies a passage from a grammatical derivation to a mathematical expression that computes a representation of the meaning of a sentence from the meanings of its words. In its most common form, this passage is expressed as a functor from a *pregroup grammar* (Lambek, 2008) to the category of finite-dimensional vector spaces and linear maps, $\mathbf{FdVect}$, where the meanings of words live in the form of vectors and tensors (Kartsaklis et al., 2016). The job of the functor is to take a grammatical derivation and translate it into a linear-algebraic operation between tensors of various orders, while the composition function that returns the meaning of the sentence is tensor contraction.

The particular choice of using a pregroup grammar as the domain of this functor is based on the fact that a pregroup, just like the semantics category on the right-hand side, has a compact-closed structure, which simplifies the transition considerably. However, while this link between pregroup grammars and DisCoCat is well-motivated, it has also been proved stronger than desired, imposing some important restrictions on the framework. As a motivating example for this paper we mention the absence of any robust statistical pregroup parser (at the time of writing) that would provide the derivations for any large-scale DisCoCat experiment on sentences of arbitrary grammatical forms. As up to the time of writing (11 years after the publication of the paper that introduced DisCoCat), all experimental work related to the model is restricted to small datasets with sentences of simple fixed grammatical structures (e.g. subject-verb-object) that are provided to the system manually.

Furthermore, pregroup grammars have been proved to be weakly equivalent to context-free grammars (Buszkowski, 2001), a degree of expressiveness that it is known to be not adequate for natural language; for example Bresnan et al. (1982) and Shieber (1985) have shown that certain syntactical constructions in Dutch and Swiss-German give rise to *cross-serial dependencies* and are beyond context-freeness. While in practice these cases are quite limited, it would still be linguistically interesting to have a version of DisCoCat that is free of any restrictions with regard to its generative power.

In this paper we overcome the above problems by detailing a version of DisCoCat whose domain is *Combinatory Categorial Grammar* (CCG) (Steed-

[1] DIStributional COmpositional CATegorical.

Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science, pages 20–31
June 16th, 2021.

man, 1987, 1996). We achieve this by encoding CCG as a biclosed category, where all standard order-preserving rules of the grammar find a natural translation into biclosed diagrams. CCG rules whose purpose is to relax word ordering and allow cross-serial dependencies are encoded as special morphisms. We then define a closed monoidal functor from the biclosed category freely generated over a set of atomic types, a set of words, and the set of arrows encoding the special rules of the grammar to a compact-closed category. We show that since the category of the semantics is symmetric, the special rules that allow word permutation can be encoded efficiently using the mechanism of "swapping the wires". As we will see in Section 3, while in the past there were other attempts to represent CCG in DisCoCat using similar methods (Grefenstette, 2013), this is the first time that a complete and theoretically sound treatment is provided and implemented in practice.

By presenting a version of DisCoCat which is no longer bound to pregroups, we achieve two important outcomes. First, since CCG is shown to be a *mildly context-sensitive* grammar (Vijay-Shanker and Weir, 1994), we increase the generative power of DisCoCat accordingly; and second, due to the availability of many robust CCG parsers that can be used for obtaining the derivations of sentences in large datasets – see, for example (Clark and Curran, 2007) – we make large-scale DisCoCat experiments on sentences of arbitrary grammatical structures possible for the first time.

In fact, we demonstrate the applicability of the proposed method by using a standard CCG parser (Yoshikawa et al., 2017) to obtain derivations for all sentences in the book "Alice's Adventures in Wonderland", which we then convert to DisCoCat diagrams based on the theory described in this paper. This resource – the first in its kind – is now available to the DisCoCat community for facilitating research and experiments. Furthermore, a web-based tool that allows the conversion of any sentence into a DisCoCat diagram is available at CQC's QNLP website.[2]

2 Introduction to DisCoCat

Based on the mathematical framework of compact-closed categories and inspired by the category-theoretic formulation of quantum mechanics (Abramsky and Coecke, 2004), the compositional distributional model of Coecke et al. (2010) computes semantic representations of sentences by composing the semantic representations of the individual words. This computation is guided by the grammatical structure of the sentence, therefore at a higher level the model can be summarised as the following transition:

$$\textbf{Grammar} \Rightarrow \textbf{Meaning}$$

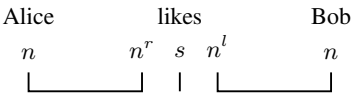
Up until now, at the left-hand side of this mapping lies a *pregroup grammar* (Lambek, 2008), that is, a partially-ordered monoid whose each element p has a left (p^l) and a right (p^r) adjoint such that:

$$p \cdot p^r \leq 1 \leq p^r \cdot p \qquad p^l \cdot p \leq 1 \leq p \cdot p^l \tag{1}$$

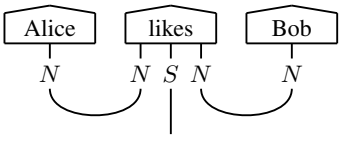
The inequalities above form the production rules of the grammar. As an example, assume a set of atomic types $\{n, s\}$ where n is a noun or a noun phrase and s a well-formed sentence, and type-assignments (Alice, n), (Bob, n), and $(\text{likes}, n^r \cdot s \cdot n^l)$; based on Eq. 1, the pregroup derivation for the sentence "Alice likes Bob" becomes:

$$n \cdot n^r \cdot s \cdot n^l \cdot n \leq 1 \cdot s \cdot 1 \leq s \tag{2}$$

showing that the sentence is grammatical. Note that the transitive verb "likes" gets the compound type $n^r \cdot s \cdot n^l$, indicating that such a verb is something that expects an n on the left and another one on the right in order to return an s. In diagrammatic form, the derivation is shown as below:



where the "brackets" ($\sqcup$) correspond to the grammatical reductions. Kartsaklis et al. (2016) showed how a structure-preserving passage can be defined between a pregroup grammar and the category of finite-dimensional vector spaces and linear maps ($\mathbf{FdVect}$), by sending each atomic type to a vector space, composite types to tensor products of spaces and cups to inner products. The DisCoCat diagram (also referred to as a *string diagram*) for the above derivation in $\mathbf{FdVect}$ becomes:



where N, S are vector spaces, "Alice" and "Bob" are represented by vectors in N, while "likes" is a tensor of order 3 in $N \otimes S \otimes N$. Here the "cups" ($\cup$)

[2] Links for the the corpus and the web demo are provided in Section 10.

correspond to *tensor contractions*, so that the vector for the whole sentence lives in S. The preference for using a pregroup grammar in the DisCoCat model becomes clear when we notice the structural similarity between the two diagrams above, and how closely the pregroup derivation dictates the shapes of the tensors and the contractions.

3 Related work

Implementations of the DisCoCat model have been provided by Grefenstette and Sadrzadeh (2011) and Kartsaklis et al. (2012), while Piedeleu et al. (2015) detail a version of the model based on density matrices for handling lexical ambiguity. DisCoCat has been used extensively in conceptual tasks such as textual entailment at the level of sentences, see for example (Bankova et al., 2019; Lewis, 2019). Further, exploiting the formal similarity of the model with quantum mechanics, Meichanetzidis et al. (2020) and Lorenz et al. (2021) have used it recently with success for the first implementations of NLP models on NISQ computers.

The connection between categorial grammars and biclosed categories is long well-known (Lambek, 1988), and discussed by Dougherty (1993). More related to DisCoCat, and in an attempt to detach the model from pregroups, Coecke et al. (2013) detail a passage from the original Lambek calculus, formed as a biclosed category, to vector spaces. In (Grefenstette, 2013) can be found a first attempt to explicitly provide categorical semantics for CCG, in the context of a functor from a closed category augmented with swaps to $\mathbf{FdVect}$. In that work, though, the addition of swaps introduces an infinite family of morphisms that collapse the category and lead to an overgenerating grammar. Further, the actual mapping of crossed composition rules to the monoidal diagrams has flaws, as given in diagrammatic and symbolic forms – see footnote 5. We close this section by mentioning the work by Maillard et al. (2014), which describes how CCG derivations can be expressed directly as tensor operations in the context of DisCoCat, building on (Grefenstette, 2013).

4 Categorial grammars

We start our exposition by providing a short introduction to categorial grammars. A *categorial grammar* (Ajdukiewicz, 1935) is a grammatical formalism based on the assumption that certain syntactic constituents are functions applied on lower-order arguments. For example, an intransitive verb gets the type $S\backslash NP$, denoting that this kind of verb is a function that expects a noun phrase on the left in order to return a sentence, while a determiner has type NP/N – a function that expects a noun on the right to return a noun phrase. The direction of the slash determines the exact position of the argument with regard to the word that represents the function. In the following derivation for the sentence "Alice likes Bob", the noun phrases and the transitive verb are assigned types NP and $(S\backslash NP)/NP$ respectively.

$$\dfrac{\dfrac{\overline{NP} \quad \overline{(S\backslash NP)/NP} \quad \overline{NP}}{S\backslash NP}>}{S}<$$

Alice likes Bob

As the diagram shows, a term with type X/Y takes a term of type Y on the right in order to return a term of type X. Similarly, a term with type $X\backslash Y$ takes a term of type Y on the left, in order to return a term of type X. In this paper we adopt a slightly different and hopefully more intuitive notation for categorial types: X/Y becomes $X \leftarrowtail Y$ while for $X\backslash Y$ we will use $Y \rightarrowtail X$. Using the new notation, the above diagram takes the form shown in Figure 1.

$$\dfrac{\dfrac{\overline{NP} \quad \overline{(NP \rightarrowtail S) \leftarrowtail NP} \quad \overline{NP}}{NP \rightarrowtail S}>}{S}<$$

Alice likes Bob

Figure 1

The two rules described above are called *forward* and *backward application*, respectively, and formally can be defined as below:

$$\text{FA } (>) \quad \dfrac{\alpha : X \leftarrowtail Y \qquad \beta : Y}{\alpha\beta : X}$$

$$\text{BA } (<) \quad \dfrac{\alpha : Y \qquad \beta : Y \rightarrowtail X}{\alpha\beta : X}$$

Categorial grammars restricted to application rules are known as *basic categorial grammars* (BCG) (Bar-Hillel, 1953), and have been proved to be equivalent to context-free grammars (Bar-Hillel et al., 1960) and pregroup grammars (Buszkowski, 2001). Interestingly, although all grammars mentioned above are equivalent in terms of theoretical expressiveness, BCGs are restrictive on the order of the reductions in a sentence. In the derivation of Figure 1, we see for example that a transitive verb must always first compose with its object, and then with the subject.

To address this problem, some categorial grammars (including CCG) contain *type-raising* and

composition rules which, although they do not affect grammar's theoretical power, allow some additional flexibility in the order of composition. These rules can be seen of as a form of *currying*, and are discussed in more depth in Section 6.

$$\text{FC } (B_{>}) \ \frac{\alpha : X \leftarrow Y \qquad \beta : Y \leftarrow Z}{\alpha\beta : X \leftarrow Z}$$

$$\text{BC } (B_{<}) \ \frac{\alpha : Z \rightarrow Y \qquad \beta : Y \rightarrow X}{\alpha\beta : Z \rightarrow X}$$

$$\text{FTR } (T_{>}) \ \frac{\alpha : X}{\alpha : T \leftarrow (X \rightarrow T)}$$

$$\text{BTR } (T_{<}) \ \frac{\alpha : X}{\alpha : (T \leftarrow X) \rightarrow T}$$

In Figure 2 we see how type-raising (T) and composition (B) can be used to change the order of reductions in our example sentence, in a version that the verb is first composed with the subject and then with the object.

$$\dfrac{\dfrac{\dfrac{\text{Alice}}{NP} \quad \dfrac{\text{likes}}{S \leftarrow (NP \rightarrow S)}\,{>}\mathbf{T} \quad \dfrac{\text{Bob}}{NP}}{(NP \rightarrow S) \leftarrow NP}\,{>}\mathbf{B}}{S}\,{>}$$

Figure 2

Finally, in CCG composition has also a generalized version, where additional arguments (denoted below as $\$_1$) are allowed to the right of the Z category.

$$\text{GFC } (B^n_{>}) \ \frac{\alpha : X \leftarrow Y \qquad \beta : (Y \leftarrow Z) \leftarrow \$_1}{\alpha\beta : (X \leftarrow Z) \leftarrow \$_1}$$

$$\text{GBC } (B^n_{<}) \ \frac{\alpha : X \rightarrow Y \qquad \beta : (Y \rightarrow Z) \rightarrow \$_1}{\alpha\beta : (X \rightarrow Z) \rightarrow \$_1}$$

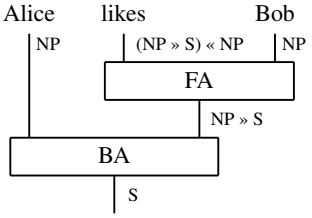
The rule can be seen as "ignoring the brackets" in the right-hand type:

$$\dfrac{\dfrac{\text{might}}{(NP \rightarrow S) \leftarrow VP} \quad \dfrac{\text{give}}{(VP \leftarrow NP) \leftarrow NP}}{((NP \rightarrow S) \leftarrow NP) \leftarrow NP}\,{>}\mathbf{B}^2$$

The generalized composition rules have special significance, since it is argued to be the reason for the beyond context-free generative capacity of CCG – see for example (Kuhlmann et al., 2015).

5 Categorial grammars as biclosed categories

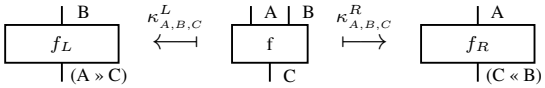
Categorial grammars can be seen as a proof system, and form a *biclosed category* $\mathcal{B}$ whose objects are the categorial types while the arrows $X \rightarrow Y$ correspond to proofs with assumption X and conclusion Y. A word with categorial type X lives in this category as an axiom, that is, as an arrow of type $I \rightarrow X$ where the monoidal unit I is the empty assumption. Below we show the biclosed diagram for the CCG derivation of the sentence "Alice likes Bob":



We remind the reader that a biclosed category is both left-closed and right-closed, meaning that it is equipped with the following two isomorphisms:

$$\kappa^L_{A,B,C} : \mathcal{B}(A \otimes B, C) \cong \mathcal{B}(B, A \rightarrow C)$$
$$\kappa^R_{A,B,C} : \mathcal{B}(A \otimes B, C) \cong \mathcal{B}(A, C \leftarrow B)$$

where κ^L corresponds to left-currying and κ^R to right-currying. Diagrammatically:



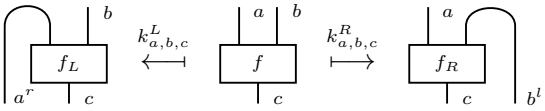
A key observation for the work in this paper is that all basic categorial rules exist naturally in any biclosed category and can emerge solely by currying and uncurrying identity morphisms; this is shown in Figure 3. Hence any CCG derivation using the rules we have met so far[3] exists in a biclosed category freely generated over atomic types and word arrows.

6 From biclosed to compact-closed

We will now define a monoidal functor from a grammar expressed as a biclosed category to DisCoCat diagrams. DisCoCat diagrams exist in a compact-closed category $\mathcal{C}$, where every object is left- and right-dualisable and the left and right internal hom-objects between objects X and Y are isomorphic to $X^r \otimes Y$ and $Y \otimes X^l$ respectively. Thus we can directly define the left and right currying isomorphisms using the dual objects:

$$k^L_{a,b,c} : \mathcal{C}(a \otimes b, c) \cong \mathcal{C}(b, a^r \otimes c)$$
$$k^R_{a,b,c} : \mathcal{C}(a \otimes b, c) \cong \mathcal{C}(a, c \otimes b^l)$$

Left and right currying in compact-closed categories get intuitive diagrammatic representations:



[3] CCG also uses a *crossed* version of composition, which is a special case and discussed in more detail in Section 7.

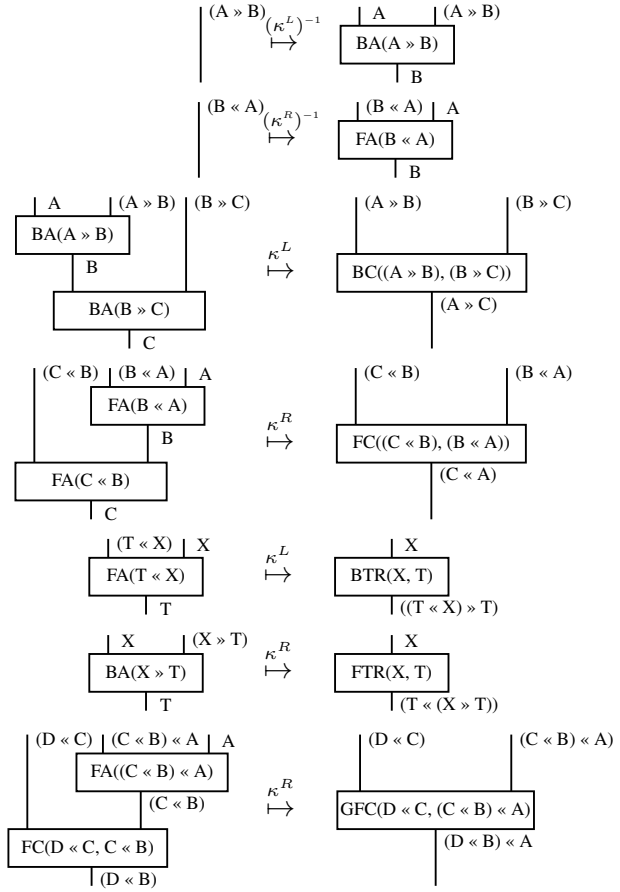


Figure 3: Categorial rules as currying/uncurrying in a biclosed category.

which allows us to functorially convert all categorial grammar rules into string diagrams, as in Figure 4.

Definition 6.1. *F is a closed monoidal functor from the biclosed category $\mathcal{B}$ of CCG derivations to the compact-closed category $\mathcal{C}$ of DisCoCat diagrams.*

Let $\{\mathrm{NP}, \mathrm{S}, \mathrm{PP}\}$ be a set of atomic CCG types, indicating a noun phrase, a sentence, and a prepositional phrase, respectively, and T a lexical type. We define the following mapping:

$$F(\mathrm{NP}) = n \quad F(\mathrm{S}) = s \quad F(\mathrm{PP}) = p$$
$$F(\mathrm{word}_{\mathcal{B}} : I_{\mathcal{B}} \to T) = \mathrm{word}_{\mathcal{C}} : I_{\mathcal{C}} \to F(T)$$

As a closed monoidal functor, $F : \mathcal{B} \to \mathcal{C}$ satisfies the following equations:

$$F(X \circ Y) = F(X) \circ F(Y) \quad F(Id_X) = Id_{F(X)}$$
$$F(X \otimes Y) = F(X) \otimes F(Y) \quad F(I_{\mathcal{B}}) = I_{\mathcal{C}}$$
$$F(X \rightarrowtail Y) = F(X)^r \otimes F(Y)$$
$$F(X \leftarrowtail Y) = F(X) \otimes F(Y)^l$$

Furthermore, for any diagram $d : A \otimes B \to C$,

$$F(\kappa^L_{A,B,C}(d)) = k^L_{a,b,c}(F(d))$$
$$F(\kappa^R_{A,B,C}(d)) = k^R_{a,b,c}(F(d))$$

where $F(A) = a, F(B) = b, F(C) = c$.

Alternatively we can say that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{B}(B, A \rightarrowtail C) & \xrightarrow{F} & \mathcal{C}(b, a^r \otimes c) \\ \uparrow{\scriptstyle \kappa^L_{A,B,C}} & & \uparrow{\scriptstyle k^L_{a,b,c}} \\ \mathcal{B}(A \otimes B, C) & \xrightarrow[F]{} & \mathcal{C}(a \otimes b, c) \\ \downarrow{\scriptstyle \kappa^R_{A,B,C}} & & \downarrow{\scriptstyle k^R_{a,b,c}} \\ \mathcal{B}(A, C \leftarrowtail B) & \xrightarrow[F]{} & \mathcal{C}(a, c \otimes b^l) \end{array}$$

As an example, below you can see how the backward application rule, derived by uncurrying an identity morphism, is converted into a string diagram in $\mathcal{C}$.

$$\begin{aligned} F(BA(A \rightarrowtail B)) &= F((\kappa^L_{A,A\rightarrowtail B,B})^{-1}(\mathrm{Id}_{A\rightarrowtail B})) \\ &= (k^L_{a,a^r\otimes b,b})^{-1}(F(\mathrm{Id}_{A\rightarrowtail B})) \\ &= (k^L_{a,a^r\otimes b,b})^{-1}(\mathrm{Id}_{F(A\rightarrowtail B)}) \\ &= (k^L_{a,a^r\otimes b,b})^{-1}(\mathrm{Id}_{a^r\otimes b}) \\ &= (k^L_{a,a^r\otimes b,b})^{-1}(\mathrm{Id}_{a^r} \otimes \mathrm{Id}_b) \end{aligned}$$

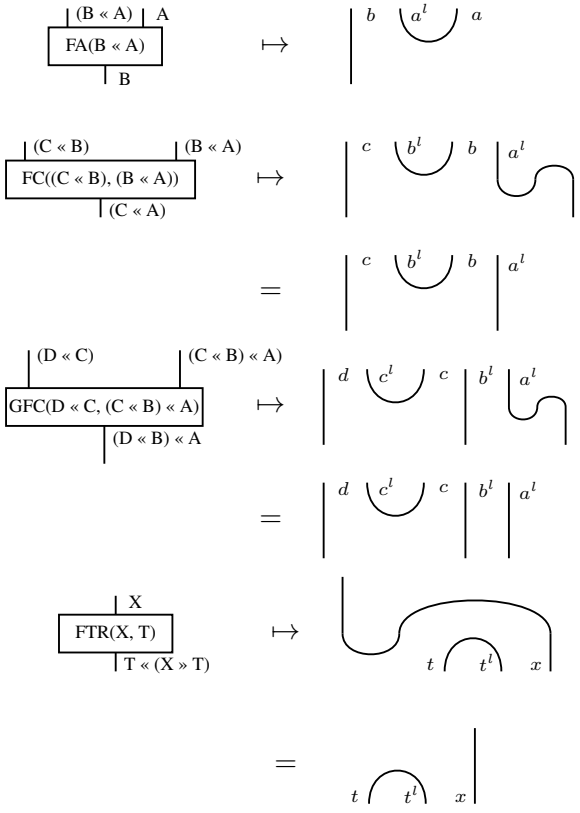


Figure 4: Functorial conversion of "forward" categorial grammar rules in biclosed form into string diagrams.

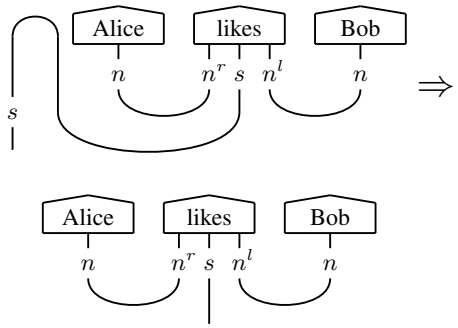


Figure 5: Rewriting of string diagrams. The starting diagram corresponds to the derivation of Figure 2, which uses type-raising. By re-arranging the sentence wire we get the non-type-raised version of Figure 1.

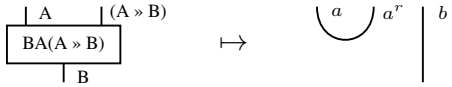
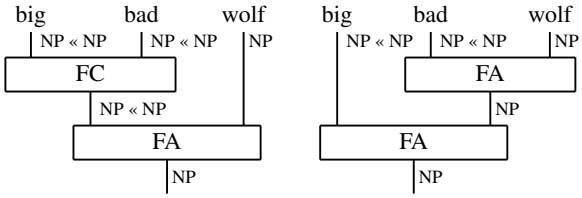
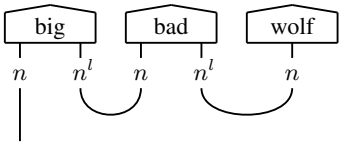


Figure 4 provides the translation of all forward rules into DisCoCat diagrams. The conversion for the backward rules can be obtained by reflecting the diagrams horizontally and replacing the left/right adjoints with right/left adjoints.

One advantage of representing parse trees using compact-closed categories over biclosed categories and categorial grammars is that the rewriting rules of string diagrams enable us to show more clearly the equivalence between two parse trees. Take for example the phrase "big bad wolf", which in biclosed form has two different derivations:



However, when these derivations are sent to a compact-closed category, they become equivalent to the following diagram which is agnostic with regard to composition order:



Another example of this is in the use of the type-raising rule in CCG, which is analogous to expansion in pregroups, and in DisCoCat can be represented using a "cap" ($\cap$). Therefore, the derivations in Figures 1 and 2, when expressed as DisCoCat diagrams, are equal up to planar isotopy (Figure 5).

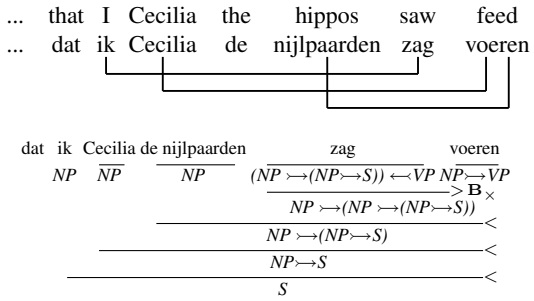


Figure 6: Cross-serial dependencies in Dutch for the phrase "...that I saw Cecilia feeding the hippos" (Steedman, 2000).

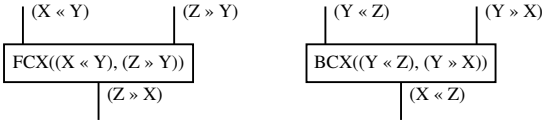
7 Crossed composition

All rules we have met so far are order-preserving, in the sense that they expect all words or textual constituents in a sentence to be found at their canonical positions. This is not always the case though, since language can be also used quite informally. To handle those cases without introducing additional types per word, CCG is equipped with the rule of *crossed composition* (Steedman, 2000), the definition of which is the following:

$$\text{FCX } (BX_{>}) \quad \frac{\alpha : X \leftarrow Y \qquad \beta : Z \rightarrow Y}{\alpha\beta : Z \rightarrow X}$$

$$\text{BCX } (BX_{<}) \quad \frac{\alpha : Y \leftarrow Z \qquad \beta : Y \rightarrow X}{\alpha\beta : X \leftarrow Z}$$

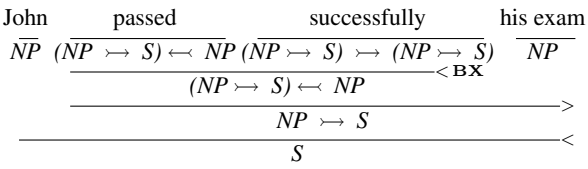
In biclosed form, the crossed composition rules are expressed as below:



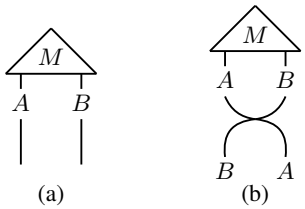
Crossed composition comes also in a generalized form as the standard (or *harmonic*) composition, and allows treatment of *cross-serial dependencies*, similar to those met in Dutch and Swiss-German (Figure 6). In English the rule is used in a restricted form[4], mainly to allow a certain degree of word associativity and permutativity when this is required.

For example, such a case is heavy *NP-shift*, where the adverb comes between the verb and its direct object (Baldridge, 2002). Consider the sentence "John passed successfully his exam", the CCG derivation of which is shown below:

[4] Steedman (2000) disallows the use of the forward version in English, while the backward version is permitted only when $Y = NP \rightarrow S$.



Note that the rule introduces a crossing between the involved types, which is not representable in pregroups. However, we remind the reader that the compact closed category where the DisCoCat diagrams live is a *symmetric* monoidal category, which means that for any two objects A and B it holds that $A \otimes B \cong B \otimes A$. In diagrammatic form this corresponds to a *swap* of the wires, as below:



In the case of **FdVect**, the state above would correspond to a matrix $M \in A \otimes B$ (a), while its swap (b) is nothing more than the transposition of that matrix, M^{T}.

Thus, by exploiting the symmetry of the semantics category, the DisCoCat diagrams for the two crossed composition rules take the form shown in Figure 7.[5]

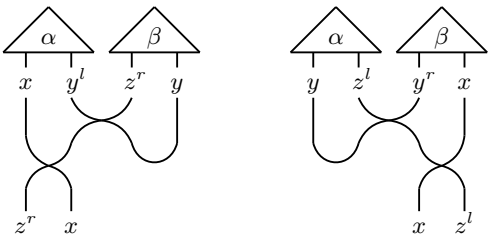
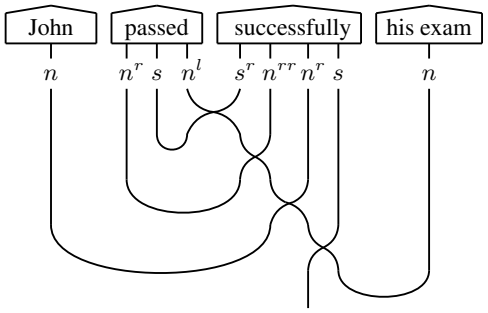


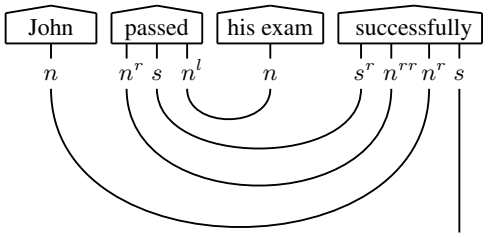
Figure 7: Crossed composition in DisCoCat (forward version on the left, backward on the right).

We are now in position to revisit the functorial passage described in Section 6 in order to include crossed composition. In contrast to other categorial rules, crossed composition does not occur naturally in a biclosed setting, so we have to explicitly add the corresponding boxes in the generating set of category $\mathcal{B}$, which is the domain of our functor. The mapping of these special boxes to compact-closed diagrams is defined in Figure 7. Deriving the generalized versions of the rules in biclosed and compact-closed form similarly to the harmonic cases is a straightforward exercise.

Based on the above, our NP-shift case gets the following diagrammatic representation:



Interestingly, this diagram can be made planar by relocating the state of the object in its canonical (from a grammar perspective) position:



which demonstrates very clearly that, in a proper use of English, permutation-inducing rules become redundant.

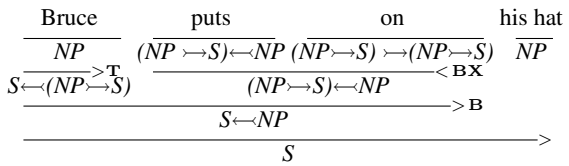
We would like to close this section with a comment on the presence of swaps in the DisCoCat category, and what exactly the implications of this are. Obviously, an unrestricted use of swaps in the semantics category would allow every possible arbitrary permutation of the words, resulting in an overgenerating model that is useless for any practical application. However, as explained in Section 2, DisCoCat is not a grammar, but a *mapping* from a grammar to a semantics. Hence it is always responsibility of the grammar to pose certain restrictions in how the semantic form is generated. In the formulation we detailed in Sections 6 and 7, we have carefully defined a biclosed category as to not introduce extra morphisms to CCG, and a functor that maps to a subcategory of a compact-closed category such that the rigid structure of traditional DisCoCat is preserved.

8 Putting everything together

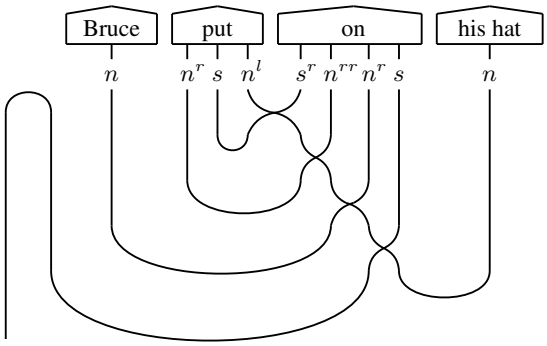
At this point we have the means to represent as a DisCoCat diagram every sentence in English language. In the following example, we consider

[5] The idea of representing crossing rules using swaps also appears in (Grefenstette, 2013); however the mapping provided there is incorrect, since there is a swap clearly missing before the last evaluation in the monoidal diagrams (p. 142, Fig. 7.7) as well as from the symbolic representations of the morphisms (p. 145).

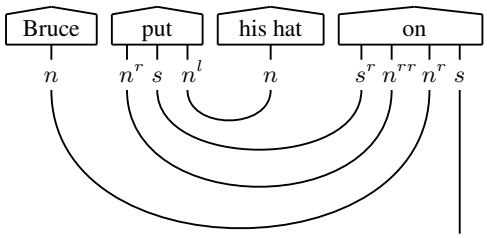
a derivation that includes type-raising, harmonic composition, and crossed composition:



The corresponding DisCoCat diagram is given below:



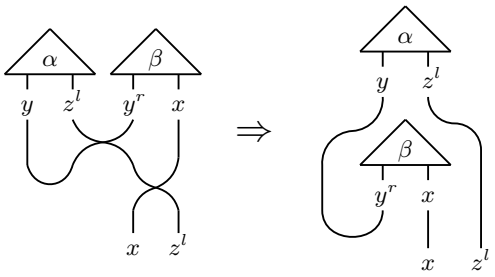
As before, relocating the object and yanking the wires produces a planar version of the diagram:



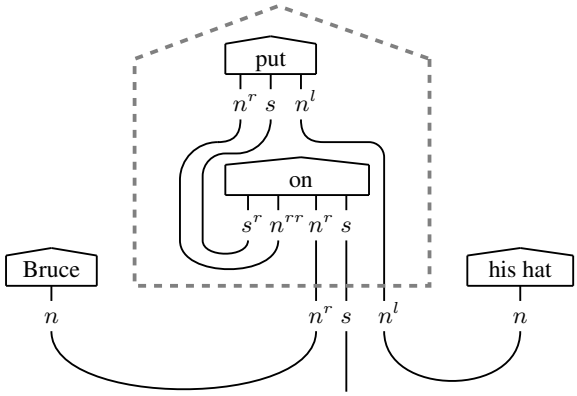
reflecting how the sentence would look if one used the separable[6] version of the phrasal verb.

9 Adhering to planarity

We have seen in Sections 7 and 8 how diagrams for sentences that feature crossed composition can be rearranged to equivalent diagrams that show a planar derivation. It is in fact *always* possible to rearrange the diagram of a derivation containing crossed composition into a planar diagram, since every instance of crossed composition between two subtrees α and β is subject to the following transformation:



By performing this rearrangement recursively on the subtrees, we obtain a planar monoidal diagram for the whole derivation. For example, a sentence containing a phrasal verb gets the following diagram:



Note how the two constituents of the phrasal verb are grouped together in a single state with type $n^r \cdot s \cdot n^l$, forming a proper transitive verb, and how the diagram is planar by construction without the need of any rearrangement.

Being able to express the diagrams without swaps is not only linguistically interesting, but also computationally advantageous. As mentioned before, on classical hardware swaps correspond to transpositions of usually large tensors; on quantum hardware, since a decomposition of a swap gate contains entangling gates, by reducing the number of swaps in a diagram we reduce the currently expensive entangling gates (such as CNOTs) required to synthesise the diagram.

10 A DisCoCat version of "Alice in Wonderland"

We demonstrate the theory of this paper by converting Lewis Carroll's "Alice in Wonderland"[7] in DisCoCat form. Our experiment is based on the following steps:

[6] A phrasal verb is *separable* when its object can be positioned between the verb and the particle.

[7] We used the freely available version of Project Gutenberg (https://www.gutenberg.org).

1. We use DepCCG parser[8] (Yoshikawa et al., 2017) to obtain CCG derivations for all sentences in the book.

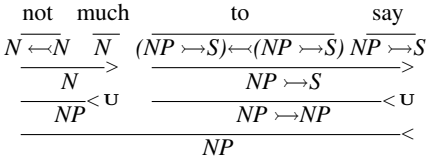
2. The CCG derivation for a sentence is converted into biclosed form, as described in Section 5.

3. Finally, the functorial mapping from biclosed to string diagrams is applied, as detailed in Sections 6 and 7.

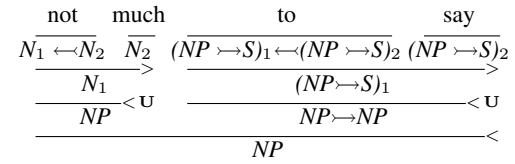
The DepCCG parser failed to parse 18 of the 3059 total sentences in the book, resulting in a set of 3041 valid CCG derivations, all of which were successfully converted into DisCoCat diagrams based on the methodology of this paper. The new corpus is now publicly available to further facilitate research in DisCoCat[9], and is provided in three formats: biclosed, monoidal, and DisCoCat, while PDF versions of the diagrams are also available. For the representation of the diagrams we used DisCoPy[10] (de Felice et al., 2020), a Python library for working with monoidal categories. Further, a Web tool that allows the conversion of any sentence to DisCoCat diagram providing various configuration and output options, including LaTeX code for rendering the diagram in a LaTeX document, is available at CQC's website[11]. In the Appendix we show the first few paragraphs of the book in DisCoCat form by using this option.

11 Some practical considerations

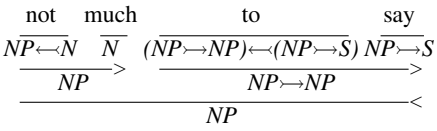
For the sake of a self-contained manuscript, in this section we discuss a few important technicalities related to CCG parsers that cannot be covered by the theory. The most important is the concept of *unary rules*, where a type is changed in an ad-hoc way at some point of the derivation in order to make an outcome possible. In the following CCG diagram, we see unary rules (U) changing $NP \rightarrowtail S$ to $NP \rightarrowtail NP$ and N to NP at a later point of the derivation.



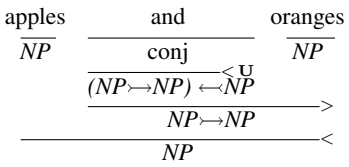
We address this problem by employing an indexing system that links the categorial types with their corresponding arguments in a way that is always possible to traverse the tree backwards and make appropriate replacements when a unary rule is met. For the above example, we get:



Applying the unary rules is now distilled into replacing all instances of N_1 with NP and $(NP \rightarrowtail S)_1$ with $NP \rightarrowtail NP$ in the already processed part of the tree, which leads to the following free of unary rules final diagram:



Finally, we discuss conjunctions, which in CCG parsers take the special type *conj*. We essentially treat these cases as unary rules, constructing the destination type by the types of the two conjuncts:



12 Future work and conclusion

In this paper we showed how CCG derivations can be expressed in DisCoCat, paving the way for large-scale applications of the model. In fact, presenting a large-scale experiment based on DisCoCat is a natural next step and one of our goals for the near future. Creating more DisCoCat-related resources, similar to the corpus introduced in this paper, is an important direction with obvious benefits to the community.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. We are grateful to Steve Clark for his comments on CCG and the useful discussions on the generative power of the formalism. The paper has also greatly benefited from discussions with Alexis Toumi, Vincent Wang, Ian Fan, Harny Wang, Giovanni de Felice, Will Simmons, Konstantinos Meichanetzidis and Bob Coecke, who all have our sincere thanks.

[8] https://github.com/masashi-y/depccg

[9] https://qnlp.cambridgequantum.com/downloads.html.

[10] https://github.com/oxford-quantum-group/discopy

[11] https://qnlp.cambridgequantum.com/generate.html

References

S. Abramsky and B. Coecke. 2004. A Categorical Semantics of Quantum Protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425. IEEE Computer Science Press. arXiv:quant-ph/0402130.

Kazimierz Ajdukiewicz. 1935. Die syntaktische konnexitat. *Studia philosophica*, pages 1–27.

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh, School of Informatics.

Dea Bankova, Bob Coecke, Martha Lewis, and Dan Marsden. 2019. Graded Entailment for Compositional Distributional Semantics. *Journal of Language Modelling*, 6(2):225–260.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29(1):47–58.

Yehoshua Bar-Hillel, Gaifman (C.), Eli Shamir, and C Caifman. 1960. *On categorial and phrase-structure grammars*. Weizmann Science Press.

Joan Bresnan, Ronald M Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial Dependencies in Dutch. In *The formal complexity of natural language*, pages 286–319. Springer.

Wojciech Buszkowski. 2001. Lambek grammars based on pregroups. In *International Conference on Logical Aspects of Computational Linguistics*, pages 95–109. Springer.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh. 2013. Lambek vs. Lambek: Functorial Vector Space Semantics and String Diagrams for Lambek Calculus. *Annals of Pure and Applied Logic*, 164(11):1079–1100. Special issue on Seventh Workshop on Games for Logic and Programming Languages (GaLoP VII).

Bob Coecke, Mehrnoosh Sadrzadeh, and Steve Clark. 2010. 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.

Daniel J. Dougherty. 1993. Closed Categories and Categorial Grammar. *Notre Dame journal of formal logic*, 34(1):36–49.

Giovanni de Felice, Alexis Toumi, and Bob Coecke. 2020. DisCoPy: Monoidal Categories in Python. In *Proceedings of the 3rd Annual International Applied Category Theory Conference*. EPTCS.

Edward Grefenstette. 2013. *Category-theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. Ph.D. thesis, University of Oxford, Department of Computer Science.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A Unified Sentence Space for Categorical Distributional-Compositional Semantics: Theory and Experiments. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 549–558.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, Stephen Pulman, and Bob Coecke. 2016. *Reasoning about meaning in natural language with compact closed categories and Frobenius algebras*, Lecture Notes in Logic, page 199–222. Cambridge University Press.

Marco Kuhlmann, Alexander Koller, and Giorgio Satta. 2015. Lexicalization and Generative Power in CCG. *Computational Linguistics*, 41(2):187–219.

J. Lambek. 2008. *From Word to Sentence*. Polimetrica, Milan.

Joachim Lambek. 1988. Categorial and Categorical Grammars. In *Categorial grammars and natural language structures*, pages 297–317. Springer.

Martha Lewis. 2019. Modelling Hyponymy for DisCoCat. In *Proceedings of the Applied Category Theory Conference*, Oxford, UK.

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 preprint arXiv:2102.12846*.

Jean Maillard, Stephen Clark, and Edward Grefenstette. 2014. A Type-Driven Tensor-Based Semantics for CCG. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 46–54, Gothenburg, Sweden. Association for Computational Linguistics.

Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, and Bob Coecke. 2020. Grammar-Aware Question-Answering on Quantum Computers.

Robin Piedeleu, Dimitri Kartsaklis, Bob Coecke, and Mehrnoosh Sadrzadeh. 2015. Open System Categorical Quantum Semantics in Natural Language Processing. In *Proceedings of the 6th Conference on Algebra and Coalgebra in Computer Science*, Nijmegen, Netherlands.

Stuart M Shieber. 1985. Evidence Against the Context-Freeness of Natural Language. In *Philosophy, Language, and Artificial Intelligence*, pages 79–89. Springer.

Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5(3):403–439.

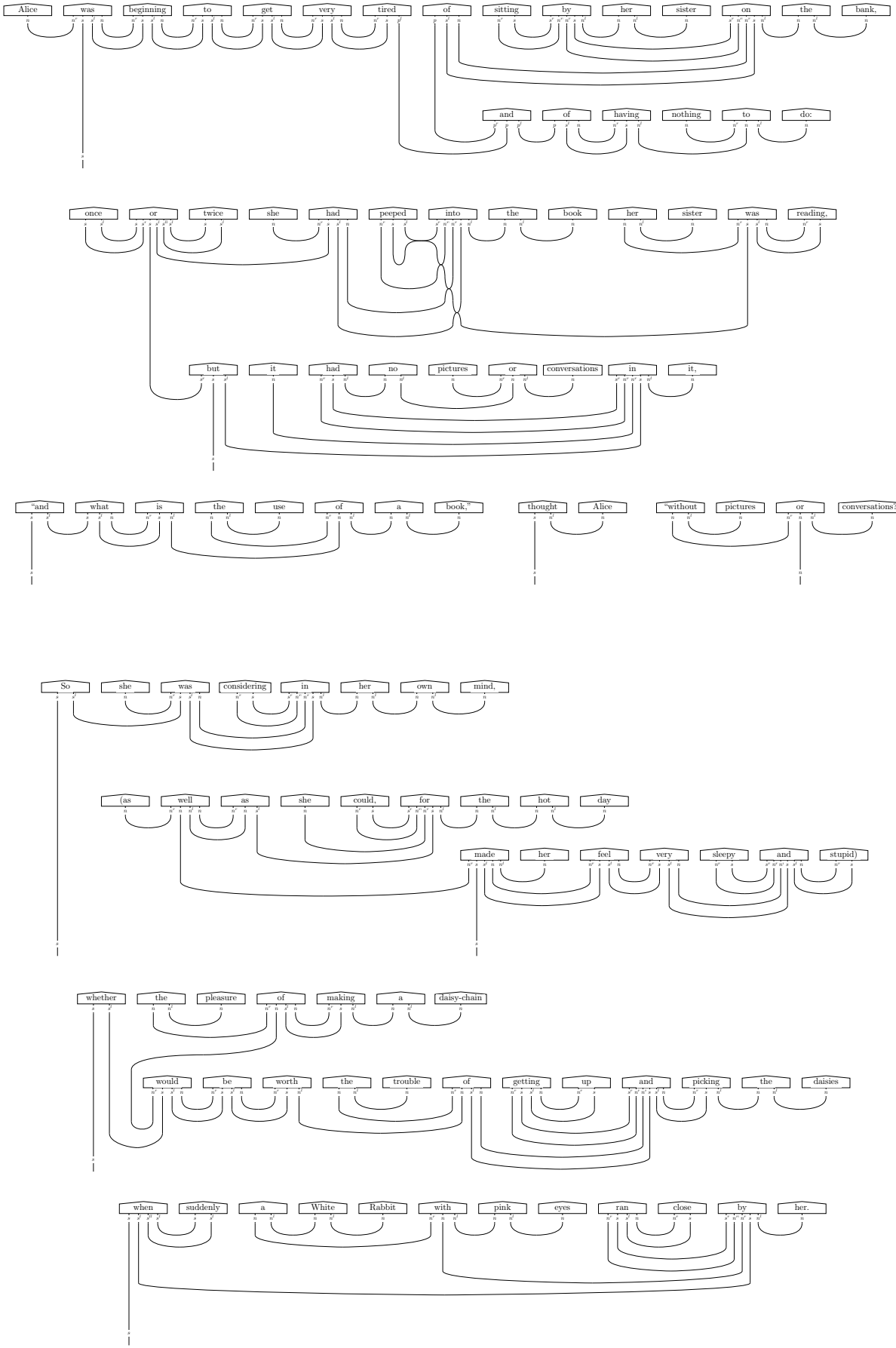
Mark Steedman. 1996. A very short introduction to ccg. *Unpublished paper. http://www. coqsci. ed. ac. uk/steedman/paper. html*.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Krishnamurti Vijay-Shanker and David J Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical systems theory*, 27(6):511–546.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* ccg parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287. Association for Computational Linguistics.

A Appendix


Alice was beginning to get very tired of sitting by her sister on the bank,
and of having nothing to do:
once or twice she had peeped into the book her sister was reading,
but it had no pictures or conversations in it,
“and what is the use of a book,”
thought Alice
“without pictures or conversations?”
So she was considering in her own mind,
(as well as she could, for the hot day
made her feel very sleepy and stupid)
whether the pleasure of making a daisy-chain
would be worth the trouble of getting up and picking the daisies
when suddenly a White Rabbit with pink eyes ran close by her.