Cambridge Quantum

Quantum Natural Language Processing on Near-Term Quantum Computers

by K. Meichanetzidis, G. De Felice, A. Toumi, B. Coecke, S. Gogioso, N. Chiappori

Cambridge Quantum
Department of Computer Science
University of Oxford, United Kingdom
KONSTANTINOS MEICHANETZIDIS
konstantinos.meichanetzidis@cs.ox.ac.uk
GIOVANNI DE FELICE
giovanni.defelice@cs.ox.ac.uk
ALEXIS TOUMI
alexis.toumi@cs.ox.ac.uk
BOB COECKE
bob.coecke@cs.ox.ac.uk

Hashberg Limited, United Kingdom
STEGANO GOGIOSO
stefano.gogioso@hashberg.io
NICOLO CHIAPPORI
nicolo.chiappori@hashberg.io

Cambridge Quantum
Terrington House
13-15 Hills Road
Cambridge CB2 1NL
United Kingdom

Published by Cambridge Quantum

08 May 2020

Quantum Natural Language Processing on Near-Term Quantum Computers

K. Meichanetzidis, G. De Felice, A. Toumi and B. Coecke
Cambridge Quantum Computing
Department of Computer Science, University of Oxford
{konstantinos.meichanetzidis, giovanni.defelice, alexis.toumi, bob.coecke}(at)cs.ox.ac.uk

S. Gogioso and N. Chiappori
Hashberg
{stefano.gogioso, nicolo.chiappori}(at)hashberg.io

In this work, we describe a full-stack pipeline for natural language processing on near-term quantum computers, aka QNLP. The language modelling framework we employ is that of compositional distributional semantics (DisCoCat), which extends and complements the compositional structure of pregroup grammars. Within this model, the grammatical reduction of a sentence is interpreted as a diagram, encoding a specific interaction of words according to the grammar. It is this interaction which, together with a specific choice of word embedding, realises the meaning (or "semantics") of a sentence. Building on the formal quantum-like nature of such interactions, we present a method for mapping DisCoCat diagrams to quantum circuits. Our methodology is compatible both with NISQ devices and with established Quantum Machine Learning techniques, paving the way to near-term applications of quantum technology to natural language processing.

1 Introduction

In recent years, research has flourished in the rapidly emerging fields of quantum machine learning and quantum artificial intelligence [37, 14, 13, 41]. These terms cover a vast range of topics, from consideration of agent-environment interaction in the quantum domain to potential gains in using quantum devices as subroutines for machine learning algorithms. For the purposes of this work, quantum machine learning will refer to supervised or unsupervised machine learning employing *variational quantum circuits* in place of deep neural networks [3]. The study of variational quantum circuits is important in itself, as they constitute the setting for quantum computational supremacy experiments in the current era of noisy intermediate scale quantum (NISQ) devices [33]. In this work, we focus on natural language processing (NLP), a sub-field of machine learning covering a diverse interdisciplinary landscape. Our contribution to the field will be the introduction of a framework for quantum natural language processing (QNLP), tailored for implementation on NISQ devices.

We consider the distributional-compositional models of meaning (DisCoCat) for natural language [11], mediating between the rule-based approaches to language syntax and the statistical approach to language semantics, most famously associated with John Rupert Firths assertion that *"You shall know a word by the company it keeps"*. In DisCoCat, structure is introduced via a compositional grammar model, that of *pregroup grammar*, which is then endowed with a "distributional" embedding of words into a vector space, where vector geometry captures the correlations between words according to some corpus. The interplay between compositionality of the grammar and the distributional word representation gives

Submitted to:
QPL 2020

rise to semantics for phrases and sentences: starting from embeddings for individual words (extracted from a corpus), the compositional structure of grammar makes it is possible to give meaning to larger syntactic units [27]. Tasks such as concept similarity or question answering can be then be straightforwardly translated into geometric questions about vectors and tensors, and solved computationally.

Compositional grammar models—such as pregroup grammars and context free grammars (CFG)—have a natural tensor structure [18, 31, 21] and can be considered quantum-native [42, 2, 7]. Building on the recent proposal of quantum algorithms for NLP task by Zeng and Coecke [42], we take advantage of the tensor structure in order to construct a map from DisCoCat models to variational quantum circuits, where ansätze corresponding to lexical categories—aka parts-of-speech (POS)—are connected according to the grammar to form circuits for arbitrary syntactic units. In some of its applications, the original Zeng-Coecke algorithm relies on the existence of a quantum random access memory (QRAM) [20], which is not yet known to be efficiently implementable in the absence of fault tolerant scalable quantum computers [1, 6]. Here we take a different approach, using the classical ansätz parameters to encode the distributional embedding and avoiding the need for QRAM entirely. The cost function for the parameter optimisation is informed by a corpus, already parsed and POS-tagged by classical means. Taken all together, the pipeline is as follows:

$$\sigma \in K \quad \overset{\text{parser}}{\longmapsto} \quad D \in \mathbf{G} \quad \overset{\text{simplify}}{\longmapsto} \quad D' \quad \overset{\text{ansatz}}{\longmapsto} \quad \mathbf{QCirc} \quad \overset{\text{compiler}}{\longmapsto} \quad \text{(NIS)QDev} \tag{1}$$

In the pipeline, a POS-tagged sentence σ in a corpus K is first parsed to a diagram D capturing its grammatical structure, which is further simplified to some other diagram D' more suitable for implementation. The simplified diagram is then turned into a variational quantum circuit, which is finally compiled for NISQ devices. This can be done by state-of-the art compilers, such as CQC's t|ket⟩ which is a platform-agnostic compiler and interfaces with current NISQ architectures [38]. A python wrapper for can be found at `github.com/CQCL/pytket`.

Beyond the fact that variational quantum circuits are amenable to implementation on existing NISQ hardware, a reason for constructing such variational embeddings is to exploit an entirely novel feature space in which to encode the distributional semantics [23]. Quantum-enhanced feature spaces provide a dimension which is exponential in the number of qubits, so that QNLP models have the potential to take advantage of the space for data-intensive tasks. Furthermore, the optimisation landscapes spanned by the variational quantum circuits are of different shape than those appearing in artificial neural networks, so there is the possibility for alternate performance profiles over equivalent benchmark tasks.

Contributions

This work was originally commissioned by Cambridge Quantum Computing (CQC) and was carried out independently by the CQC team and the Hashberg team.

2 From Sentence to Diagram

For the purposes of constructing our QNLP mode, we build on work which uses *pregroup grammars*, but context-free grammars (CFS) would be equally suitable for our construction. Note that pregroup grammars are weakly equivalent to CFGs [5]. To construct DisCoCat models of meaning, diagrams encoding the pregroup grammatical structure are constructed directly inside of compact closed categories—a special case of rigid categories—giving semantics to the model.

Specifically, the diagrams in this work represent complex matrices, i.e. they live in the compact closed category **fHilb**. Each atomic pregroup type is associated a finite-dimensional Hilbert space, each individual (typed) word is associated a pure state in the Hilbert associated to its type, and the pregroup grammatical structure is realised as an interaction between the word states mediated by certain entangling effects.

2.1 Compositionality by Grammar

Definition 2.1. A *pregroup* **P** is the rigid category (with chosen duals) freely generated by a finite set T of *atomic types*. Specifically, the objects in **P** are generated from T as follows:

- every atomic type $\tau \in T$ is a type (aka object) in **P**;
- for every type τ in **P**, the *left adjoint* τ^l and the *right adjoint* τ^r are also types in **P**;
- for every pair of types $\tau, \sigma \in P$, the *product type* $\tau \otimes \sigma$ is also a type **P** (typically written $\tau\sigma$);
- the product operation is strictly associative and has a bilateral unit, the *unit type* $\varepsilon \in P$;

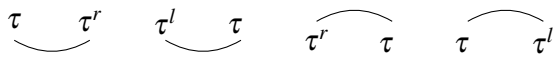
The pregroup **P** is a poset category, i.e. every pair of types A, B in T as at most one morphism $A \to B$. As convention in poset categories, we write $A \leq B$ for the unique morphism $A \to B$, if it exists. The morphisms of **P** are generated as follows:

- for every type τ in **P**, we have morphisms $\tau\tau^r \leq \varepsilon$ and $\tau^l\tau \leq \varepsilon$, known as *contractions* or *caps*;
- for every type τ in **P**, we have morphisms $\varepsilon \leq \tau^r\tau$ and $\varepsilon \leq \tau\tau^l$, known as *expansions* or *cups*;

All further equalities between objects and between morphisms follow from the requirement that **P** be a poset category.

Remark 2.2. Some useful equalities which can be derived from the requirement that **P** be a poset category include: the *snake equations* between caps and caps; the cancellation of left and right duals, $(\tau^l)^r = \tau = (\tau^r)^l$; the stability of the unit type under duals, $\varepsilon^l = \varepsilon = \varepsilon^r$; the interplay between duals and products, $(\tau\sigma)^l = \sigma^l\tau^l$ and $(\tau\sigma)^r = \sigma^r\tau^r$.

We use a graphical calculus for autonomous categories to depict morphisms in a pregroup (which are also known as *reductions*, following the tradition of rule-based grammar). In particular, the contractions and expansions are depicted as follows:



(2)

In our diagrams, objects are multiplied left-to-right and morphisms are composed top-to-bottom: in the above, the two morphism on the left are the caps/contractions and the two morphisms on the right are the cups/expansions. The empty type is the tensor unit, hence it is not depicted.

Definition 2.3. Given a pregroup **P**, a *pregroup grammar* **G** for **P** a pair $\mathbf{G} = (L, t)$ consisting of a *lexicon* L (a finite set of *words* $w \in L$) together with a *typing map* $t : L \to \mathbf{P}$ associating a pregroup type $t(w) \in P$ to each word $w \in L$.

When working with pregroup grammars, pregroup types subsume the role traditionally played by lexical categories such as nouns, adjectives, verbs, adverbs, etc [27]. If the lexicon is already POS-tagged by other means, then a pregroup grammar can be obtained by associating pregroup types to each POS-tag. A pregroup grammar **G** can equivalently be seen as the strict monoidal category generated from **P** by

adding states $\varepsilon \to \tau$ labelled by the words $\{w \in L \,|\, t(w) = \tau\}$ for each individual type $\tau \in P$. This is the same as saying that $\mathbf{G}$ is the rigid category (with chosen duals) freely generated by the atomic types of $\mathbf{P}$ and by states $w : \varepsilon \to t(w)$ for all words $w \in L$.

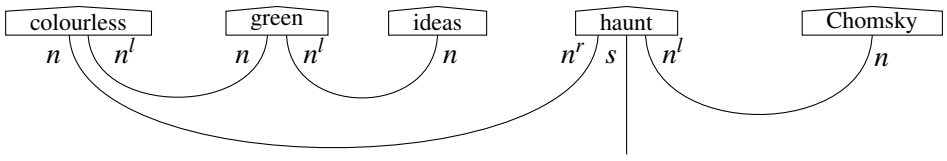
Remark 2.4. Taking the left/right duals gives monoidal functors $(\_)^l, (\_)^r : \mathbf{P} \to \mathbf{P}^{op}$ and by extension monoidal functors $(\_)^l, (\_)^r : \mathbf{G} \to \mathbf{G}^{op}$, where the opposite categories $\mathbf{P}^{op}$ and $\mathbf{G}^{op}$ (those with arrows reversed) are equipped with the opposite monoidal product $A \boxtimes B := B \otimes A$ with respect to the original categories $\mathbf{P}$ and $\mathbf{G}$.

Definition 2.5. Consider a pregroup $\mathbf{P}$ with atomic types T is equipped with a chosen *sentence type* $s \in T$. A *grammatical sentence* is a non-empty sequence $\underline{w} = (w_1, ..., w_n)$ of words together with a sequence of contractions and expansions witnessing that the product type associated to the sequence reduces to the sentence type:

$$\prod_{i=1}^{n} t(w_i) \leq s \tag{3}$$

In this work, empty products are identified with the unit type ε and non-empty products are expanded left-to-right as $\prod_{i=1}^{n} t(w_i) := t(w_1) \dots t(w_n)$. [1]

Deciding *grammaticality*—that is deciding whether the product type associated to a given sequence of words by a pregroup grammar $\mathbf{G}$ reduces to the chosen sentence type s—is an efficiently solvable problem [32, 16]. In the graphical calculus, the witness of grammaticality for a sequence of words is a pattern of nested caps (and possibly cups) connecting the atoms in the product type in such a way as to leave only one s type open:



(4)

There exist several measures of how grammatical a sentence is, such as *Harmony*. For pregroup grammars, Harmony can be defined as the number of non-sentence atomic types left open after parsing [29]. Harmony maximisation—i.e. finding the parsing closest to a witness of sentence grammaticality—non-trivial problem which enjoys polynomial quantum speed-up [39].

2.2 Distributional Meaning

Given a pregroup grammar $\mathbf{G}$, semantics for the grammar are given by monoidal functors $F : \mathbf{G} \to \mathbf{C}$, where $\mathbf{C}$ is some suitable rigid category (with compact closed categories as a special case). In *distributional semantics*, dagger compact categories of finite-dimensional Hilbert spaces are often of interest, such as the category **fHilb** of complex matrices used in this work or its real matrices analogue, used in most traditional approaches to NLP. There is a vast literature on methods for associating distributional semantics to words, from the early bag-of-words approaches to more modern ones based on artificial

[1] Such a choice of convention is made necessary by non-commutativity of the product operation on types.

neural networks [30]. Non-vectorial representations have also appeared in the literature [4]. Compositional distributional semantics has been successfully benchmarked against more traditional approaches, outperforming several of the contemporary techniques [22, 24, 25, 40].

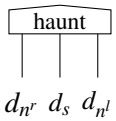
We work in a presentation of **fHilb** where objects are the positive integers—the possible dimensions for finite-dimensional Hilbert spaces—and morphisms $m \to n$ are n-by-m complex matrices. The presentation is made dagger compact by taking the conjugate transpose of matrices as the dagger, together with the following chosen duals:

1. we pick our dual objects as $n^* := n$;
2. we make a choice of orthonormal basis—the *computational basis*—for all n prime;
3. we extend our chosen computational bases to all n by considering product bases;
4. we define the cap $n \otimes n \to 1$ by setting $|e_i\rangle \otimes |e_j\rangle \mapsto \delta_{ij}$ on the chosen orthonormal basis for n;
5. we define the cup $1 \to n \otimes n$ as the adjoint of the cap $n \otimes n \to 1$.

With the above presentation, giving distributional semantics $\mathbf{G} \to \mathbf{fHilb}$ concretely means the following:

- a finite dimension d_τ is associated to each atomic type $\tau \in T$ of the pregroup $\mathbf{P}$; [2]
- a $d_{t(w)}$-dimensional complex vector $|w\rangle$ is associated to each word $w \in L$.

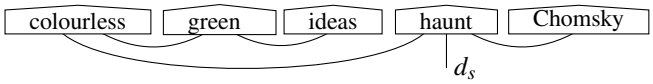
We refer to the data above as a *word embedding*. For example, the the word "haunt" would have a complex vector of dimension $d_{n^r} d_s d_{n^l}$ associated to it by a word embedding, which we can represent as follows in the graphical calculus:



(5)

Remark 2.6. Because information about the factors of the dimension $d_{n^r} d_s d_{n^l}$ is derivable from the word embedding together with the word type $t(\text{haunt})$, we can equivalently treat the above as a vector in dimension $d_{n^r} d_s d_{n^l}$ or as a tensor or arity 3 in the individual dimensions d_{n^r}, d_s and d_{n^l}. In the general, the arity of the tensor associated to a word w is the number of atomic types appearing in $t(w)$.

Given a word embedding, every pregroup grammatical parsing can be turned into tensor contraction by sending the caps and caps of **G** to the chosen caps and caps of **fHilb**, as in the following example:

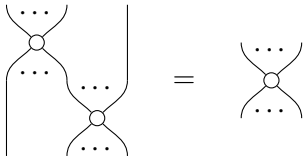


(6)

As a special case, grammatical sentences find interpretation as d_s-dimensional vectors, as the witness of grammaticality results in contraction of all tensor legs except for one of type s.

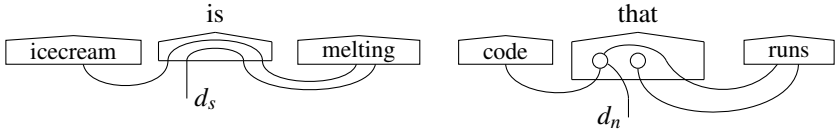
[2] Linearity and finite-dimensionality actually force the same dimension to be associated to all duals, i.e. $d_{\tau^l} = d_\tau = d_{\tau^r}$.

In our presentation of **fHilb**, each Hilbert space n has a chosen classical structure (i.e. a special commutative †-Frobenius algebra) associated to it, corresponding to our choice of computational basis. This means that spiders are available as additional ingredients to our semantics:



(7)

Spiders—with cups and caps as special two-legged cases—have been used in the past to associate semantics to functional and connective words—such as "does", "is" and "are"—or to relative pronouns—such as "which" and "that" [8, 34, 35]:



(8)

It has been argued that pregroup-based models of meaning by no means provide a complete account of linguistic phenomena. In fact, the original Lambek calculus—which pregroups later simplified—is richer and can itself instantiate a semantic model, as argued in [10]. Spiders provide one possible way of enriching such semantic models.

3 Diagram Rewriting

On the way to quantum circuits, we need to simplify our diagrams to optimise our ultimate use of quantum resources. Specifically, we present two diagram simplification methods which aim to reduce circuit width and depth independently of the choice of word ansätze. Both methods require additional flexibility in the manipulation of diagrams and hence take place in the following *symmetric* version of the pregroup grammar.

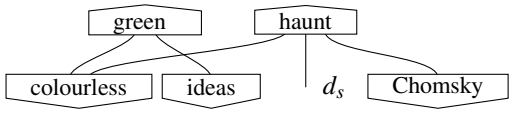
Definition 3.1. A *symmetric pregroup grammar* $\hat{\mathbf{G}}$ is the compact closed category obtained by introducing symmetry isomorphisms to a pregroup grammar $\mathbf{G}$, keeping the same objects.

If $\mathbf{G}$ is a pregroup grammar and $\hat{\mathbf{G}}$ is the associated symmetric pregroup grammar, then there is a faithful monoidal functor $i_{\mathbf{G}} : \mathbf{G} \to \hat{\mathbf{G}}$ which is the identity on objects. Any monoidal functor $F : \mathbf{G} \to \mathbf{C}$ towards a compact closed category $\mathbf{C}$ factors as $F = \hat{F} \circ i_{\mathbf{G}}$ for a unique monoidal functor $\hat{F} : \hat{\mathbf{G}} \to \mathbf{C}$. As a consequence of this observation, the introduction of a symmetric pregroup grammar provides additional degrees of freedom when it comes to diagram rewriting, without imposing any additional restrictions to the compact closed semantics traditionally considered in the DisCoCat framework.

3.1 The `bigraph` method

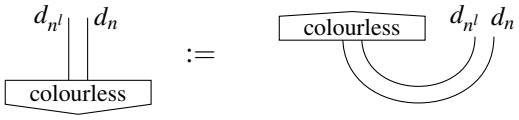
The first rewrite method we present, which we call the `bigraph` method, completes and improves the original Zeng-Coecke algorithm [42]. We start with the simplest scenario, described in the original algorithm: the diagram has a single open wire (e.g. it is a grammatical sentence) and the cups/caps connect words in such a way as to form a an acyclic (undirected) graph. For example, we could consider the grammatical sentence from (4).

As its first step, the `bigraph` method turns the diagram into a bipartite graph, based on the distance from the "root" word, which is defined to be the one connected to the unique open wire. Words of at even distance from the root are left in place as states, while words at odd distance from the root are transposed into effects:



(9)

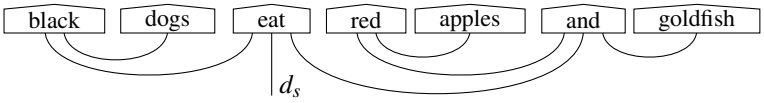
This is essentially the method originally described in [42], except that the transpose in the computational basis is used to turn states into effects, instead of the dagger used in the original formulation:



(10)

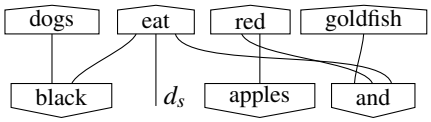
One issue not originally foreseen with this approach is the introduction of wire crossings. This is a problem when it comes to implementation on NISQ devices: a swap between neighbouring qubits involves up to three entangling gates, which in turn lead to significant increase in circuit depths and exponential decrease in fidelity.

To tackle this issue, the `bigraph` method attempts to minimise the number of crossings by altering the linear order of words in the two classes. [3] For example, consider the following grammatical sentence:



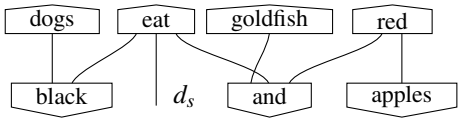
(11)

After the initial transposition step, the following bipartite graph drawing is obtained:



(12)

The drawing above involves 5 crossings: if each wire is mapped to a qubit and we use a reasonably optimised implementation of swaps between non-adjacent qubits, the crossings alone would increase the circuit depth by about 8 CNOTs. However, a simple re-ordering of the words in the two classes leads to a graph drawing involving a single crossing:



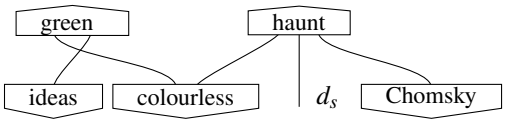
(13)

The `bigraph` method does not prescribe a specific algorithm to use when minimizing crossings: this is

[3] Because the semantic relationships between words are now encoded in the tensor contractions, their linear ordering is no longer relevant and can be used as an additional degree of freedom in the optimisation.

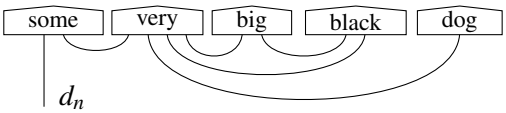
because the general problem of minimizing crossings in the planar drawing of bipartite graphs is NP-complete [19]. [4]

The `bigraph` method relies on ansätze which can be easily transposed: failing that, each transposition would naively involve a doubling of the number of qubits and the preparation or measurement of nested bell states. In Section 4 we shall see that our chosen ansätze have this property. In fact, circuit ansätze such as those used in this work often have more symmetries, which can be used to further optimise the resulting quantum circuit. For example, it is easy to transform them in such a way as to reverse the order of their outputs: this means that any combination of swaps resulting in a complete reversal of the outputs of a single word is not going to ultimately increase the depth of the quantum circuit. For example, the optimal arrangement for (9) using this additional assumption is as follows:



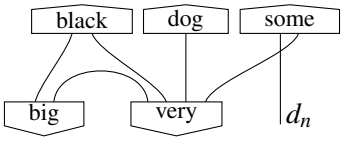
(14)

In a more general scenario, a pregroup grammatical parsing might: (i) involve cups/expansions as well as caps/contractions; (ii) result in a cyclic graph. The presence of cups/expansions is a non-issue when it comes to semantics in compact closed symmetric monoidal categories: thanks to the existence of symmetry isomorphisms, all cup will be cancelled out by caps in the target category. The case of cyclic graphs requires more careful handling. For example, consider the following odd parsing:



(15)

In the presence of cycles, distance from the root is no longer a well-defined notion and it may not be possible to re-arrange the diagram as to form a bipartite graph. Given any partition of words into two linearly ordered classes—a "pseudo-bipartite" drawing, let's call it—each edge between words of same class can be "dragged" to the other side as if it were a word, increasing the circuit width by two wires. This can be seen in the following re-arrangement for the odd parsing above:



(16)

In this more general scenario, a cost function is required by the `bigraph` method to establish a trade-off between minimising the number of crossings and minimising the number of intra-class edges in a "pseudo-bipartite" drawing of the diagram.

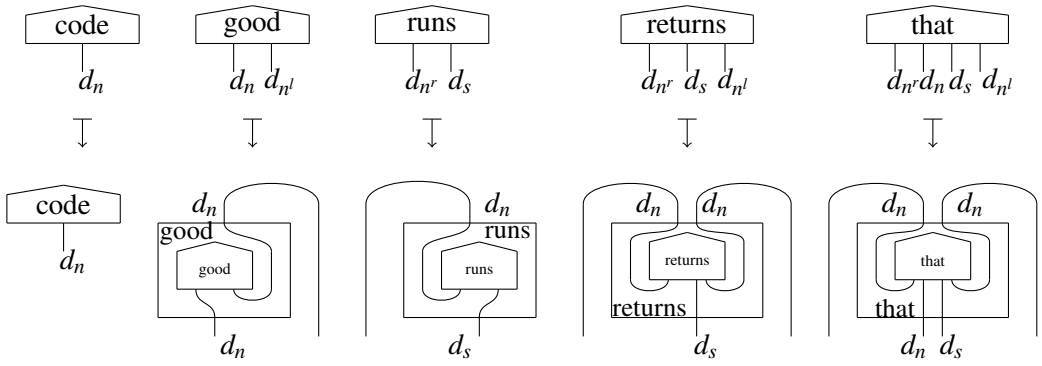
Having handled all of the above, there is a final issue to consider: when dealing with parsings other than grammatical sentences, it is not necessary (nor necessarily desirable) that a single wire be left open. To handle this most general scenario, the `bigraph` method operates as in the cyclic case—i.e. looks for a bipartite drawing optimising some trade-off between lack of crossings and lack of intra-class edges—but restricting the partitions in such a way that all words having one or more open wires are placed in the same class. This ensures that the result always be a state, as was the case so far.

[4] It is an open question whether the bipartite graphs that arise from pregroup grammars form a sub-class which is sufficiently restricted—e.g. due to the localised range of the connections—to bring the complexity down to P.

(A Python implementation of the `bigraph` method will be available at github.com/hashberg-io/qnlp.)

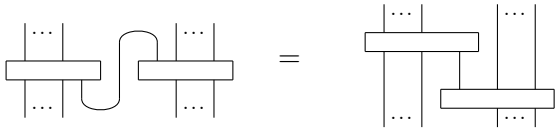
3.2 The `snake_removal` method

The second rewrite method we present, which we call the `snake_removal` method, is based on previous results by Ref. [15] and [12]. Instead of working with the full symmetric pregroup grammar $\hat{\mathbf{G}}$, the `snake_removal` method considers the full sub-category of $\hat{\mathbf{G}}$ spanned only by the atomic types and their products, which we call $\hat{\mathbf{G}}_{aut}$. This subcategory does not contain any word states which involve any adjoint types: instead, it contains the partial transposes of those states where all output wires with adjoint type have been bent into input wires. A set of generators for this sub-category can be obtained by picking one representative for each word. This procedure is called the *autonomisation* of diagrams [12] and some examples of its application can be found below:



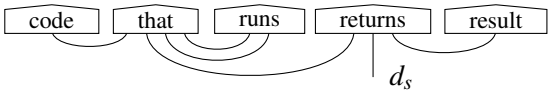
(17)

The `snake_removal` method prescribes the autonomisation of each word and subsequent yanking of the wires, as done in Def. 2.12 of Ref. [15]:



(18)

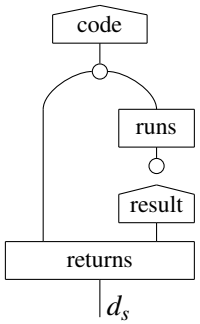
The end result is a "snake-free" diagram with no cups and caps, which can be interpreted any symmetric monoidal category. For example, consider the following grammatical sentence:



(19)

The `snake_removal` method would result in the following "snake-free" diagram (where we have used

the classical structure ansatz for "that" from (8)):



(20)

When translating the resulting snake-free diagrams into quantum circuits, it is important to note that process-state duality requires *all linear maps* to be available in the autonomisation process, not only the unitary ones. The realisation of non-unitary maps requires ancillary states and post-selection: this leads to an increase in circuit width and—ceteris paribus—an exponentially higher number of samples required during computation. If post-selection is not a viable option, then the restriction to unitary maps in turn imposes significant restrictions on the states available for words. For example, adjectives cannot change the semantic distance between the nouns the modify: a "spherical cow" and a "spherical chicken" will have the same semantic distance that the unmodified "cow" and "chicken" previously had, regardless of whether they are in a vacuum or not.

(A Python implementation of the `snake_removal` method, as part of the DisCoPy toolbox for monoidal categories, is available at github.com/oxford-quantum-group/discopy. For more details see the accompanying technical paper, Ref. [17])

4 From Diagram to Circuit

The last step in our pipeline is the association of ansätze to words, either in the form of state ansätze (for the `bigraph` method) or in the form of process ansätze (for the `snake_removal` method). We consider two generic families of unitary qubit ansätze, the CNOT+U(3) ones and the IQP ones. Each atomic type τ is mapped to one or more qubits, i.e. we have $d_\tau = 2^{n_\tau}$. More general ansätze are derived from the unitary ones as follows:

- state ansätze are obtained by application of the unitary ansätze to the Pauli Z $|0\rangle$ state;
- effect ansätze are obtained by transposition of the unitary ansätze and post-selection onto the Pauli Z measurement outcome corresponding to the $\langle 0|$ effect; [5]
- more general linear map ansätze method are obtained by using ancillary qubits prepared in the $|0\rangle$ state and/or post-selecting onto the measurement outcome corresponding to the $\langle 0|$ effect.

For the `bigraph` method, semantics $\hat{\mathbf{G}} \to \mathbf{fHilb}$ are given by associating each word to a linear map ansatz and then constructing the following monoidal functor:

- word states are mapped to the state ansätze described above;
- word effects are mapped to the effect ansätze described above;

[5] Not that the word post-selection is used here to denote the linear process where no re-normalisation of probabilities is performed.

- wire crossing are mapped to swaps, cups are mapped to preparation of a Bell state, caps are mapped to post-selection onto the measurement outcome corresponding to the same Bell state.

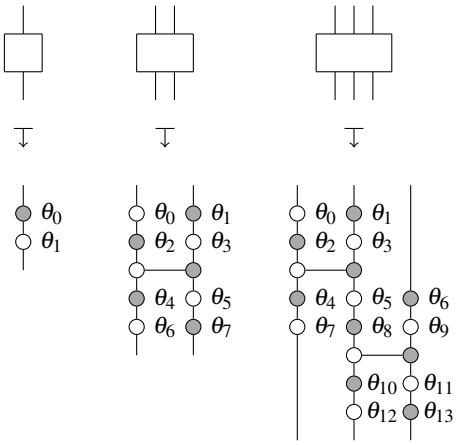
For the `snake_removal` method, semantics $\hat{\mathbf{G}}_{aut} \to \mathbf{fHilb}$ are given by associating each word to a linear map ansatz and then constructing the following monoidal functor:

- the chosen word representatives in $\hat{\mathbf{G}}_{aut}$ are mapped to the linear map ansätze;
- wire crossing are mapped to swaps;

With the exception of functional and connection words, the ansätze are parametrised. Typically we associate a single parametric ansatz to all words with the same POS, with the specific values of the parameters distinguishing between the words.

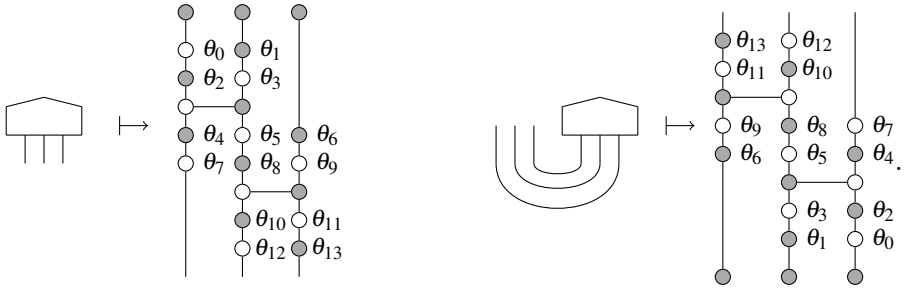
4.1 CNOT+U(3) ansätze

This family of ansätze consists of unitary quantum circuits formed by alternating layers of single-qubits rotations in X and Z with layers of CNOT gates between neighbouring qubits. The examples below—for 1, 2 and 3 qubits respectively—are written in ZX calculus notation [9]. Single-qubits white and black dots are rotations in Pauli Z and Pauli X respectively, while a black and a white dot connected by a horizontal line is a CNOT gate:



(21)

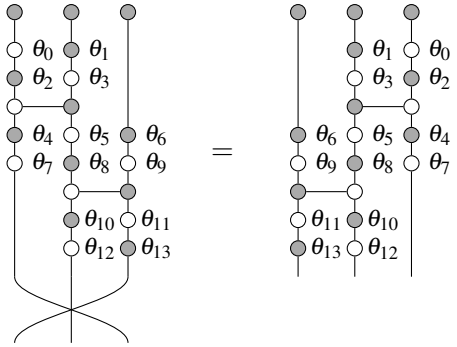
State ansätze are obtained by applying the unitary ansätze to the zero state of the computational basis, following the convention on IBMQ devices, and effect ansätze are obtained by transposing the state ansätze in the computational basis:



(22)

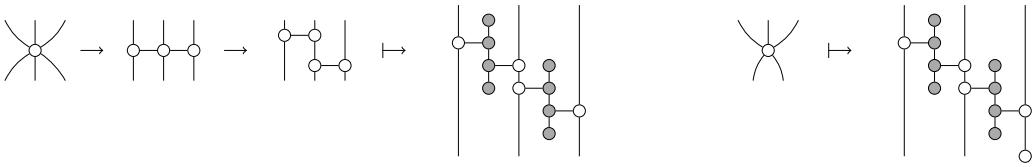
The effect is post-selection (without re-normalisation) onto the Pauli Z measurement outcome corresponding to the $\langle 0|$ effect. This family of ansätze transforms nicely under reversal of all inputs/outputs,

as shown by the following 3-qubit example:



(23)

We have said before that functional and connection words are often modelled in the DisCoCat literature using spiders [8, 34, 35]. As a consequence, it is interesting to see how spiders can be realised as non-parametric of CNOT+U(3) ansätze. Spiders with the same number of input and output legs are obtained from alternating CNOT-TONC ladders with preparation and post-selection on ancillary qubits:

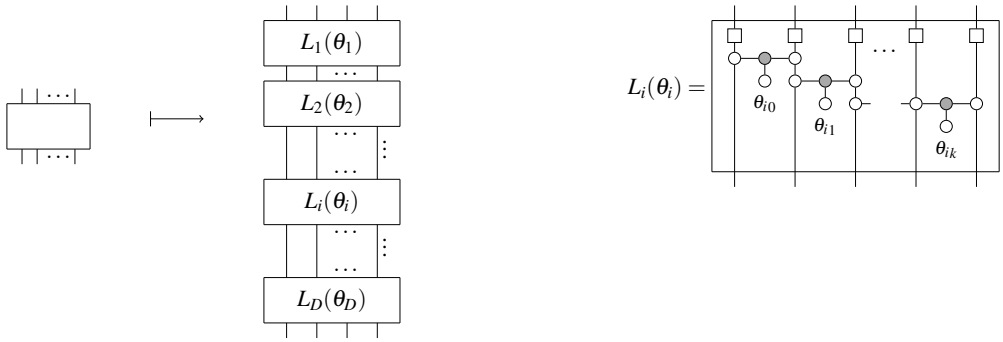


(24)

Spiders with a different number of input and output legs are then obtained by application of input legs to $|+\rangle$ states or post-selection of output legs onto the Pauli X measurement outcome corresponding to the $\langle+|$ effect.

4.2 IQP ansätze

This family of ansätze consists of instantaneous quantum polynomial (IQP) circuits. IQP circuits constitute of one or more layers, each layer consisting of a row of Hadamard gates, followed by a ladder of parametrised controlled-Z rotations—the rotations commute, hence the name "instantaneous". At the end, a final row of Hadamards is applied. Here follows a schematic representation of one such circuit:



(25)

As with the previous family, state ansätze are obtained by application to $|0\rangle$ states and effect ansätze are obtained by post-selection against $\langle 0|$ effects. Transposition of an IQP ansatz results in another IQP ansatz with layers and rotations in reverse order. Reversal of all inputs and outputs of an IQP ansatz results in another IQP ansatz with layers in the same order but rotations in reverse order.

5 Future Work

In this work, we have described a pipeline for the implementation of NLP tasks on quantum devices, by compositional translation of lexical structures to parametrised quantum circuits. We have provided two methods—named `bigraph` and `snake_removal`—for optimising the resulting circuits, developed with the goal of near-term implementation on NISQ devices.

These are humble first steps in uncharted territory and much work remains to be done on the practical side of things. Firstly, our optimisation methods are limited by the assumption that qubits be arranged linearly, and the job of making optimal use of each machine-specific arrangement is left to the transpiler of the specific device, such as IBM's, or an independent compiler, such as t$|$ket$\rangle$. Future work will take qubit topology into consideration when minimising the number of crossings in the `bigraph` method. Secondly, we intend to put a lot of work into benchmarking the optimisation algorithms, ansatz choices, training methods and various hyper-parameters, including an investigation of the relationship between corpus size, wire dimensionality and generalisation. Finally, we will explore alternative quantum computing models, such as continuous-variable, adiabatic and measurement-based.

A lot also remains to be done on the theoretical side. As an example, we note that not all linguistic phenomena are well approximated by the use of context-free grammars. Lambek himself proposed the introduction of a "meet" operation, combining two or more pregroup grammars in non–context-free way [26]. It will be interesting to investigate the various ways in which such a model can be mapped onto quantum circuits, to accommodate our choice distributional semantics of **fHilb**. Another interesting question concerns the incorporation of mixed behaviour in the semantics themselves, moving from **fHilb** to the operator model of quantum theory. Density matrices have already been used in the literature to model entailment and ambiguity [36, 28] and they can be practically realised with quantum circuits by incorporating measurements and controlled operations, with polynomial overhead.

Acknowledgements

KM and SG would like to acknowledge useful and interesting discussions with Vojtěch Havlíček and Antonin Delpeuch. KM is supported by a Research Fellowship by the Royal Commission for the Exhibition of 1851 (`royalcommission1851.org`). All diagrams were drawn with TikZiT (`tikzit.github.io`). KM, GDF, AT and BC would like to acknowledge financial support from Cambridge Quantum Computing Ltd. SG and NC would like to acknowledge financial support from Hashberg Ltd.

References

[1] Scott Aaronson (2015): *Read the fine print*. *Nature Physics* 11(4), pp. 291–293, doi:10.1038/nphys3272.

[2] Itai Arad & Zeph Landau (2010): *Quantum Computation and the Evaluation of Tensor Networks*. *SIAM Journal on Computing* 39(7), p. 30893121, doi:10.1137/080739379.

[3] Marcello Benedetti, Erika Lloyd, Stefan Sack & Mattia Fiorentini (2019): *Parameterized quantum circuits as machine learning models*. *Quantum Science and Technology* 4(4), p. 043001, doi:10.1088/2058-9565/ab4eb5.

[4] Josef Bolt, Bob Coecke, Fabrizio Genovese, Martha Lewis, Daniel Marsden & Robin Piedeleu (2016): *Interacting Conceptual Spaces*. *Electronic Proceedings in Theoretical Computer Science* 221, p. 1119, doi:10.4204/eptcs.221.2.

[5] Wojciech Buszkowski (2001): *Lambek Grammars Based on Pregroups*. In Philippe de Groote, Glyn Morrill & Christian Retoré, editors: *Logical Aspects of Computational Linguistics*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 95–109, doi:10.1007/3-540-48199-0_6.

[6] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini & Leonard Wossnig (2018): *Quantum machine learning: a classical perspective*. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474(2209), p. 20170551, doi:10.1098/rspa.2017.0551.

[7] S. Clark, B. Coecke, E. Grefenstette, S. Pulman & M. Sadrzadeh (2014): *A quantum teleportation inspired algorithm produces sentence meaning from word meaning and grammatical structure*. *Malaysian Journal of Mathematical Sciences* 8, pp. 15–25. Available at `https://arxiv.org/abs/1305.0556`.

[8] Stephen Clark, Bob Coecke & Mehrnoosh Sadrzadeh (2013): *The Frobenius Anatomy of Relative Pronouns*. In: *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, Association for Computational Linguistics, Sofia, Bulgaria, pp. 41–51. Available at `https://www.aclweb.org/anthology/W13-3005`.

[9] Bob Coecke & Ross Duncan (2011): *Interacting quantum observables: categorical algebra and diagrammatics*. 13, IOP Publishing, p. 043016, doi:10.1088/1367-2630/13/4/043016.

[10] Bob Coecke, Edward Grefenstette & Mehrnoosh Sadrzadeh (2013): *Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus*. *Annals of Pure and Applied Logic* 164(11), pp. 1079 – 1100, doi:10.1016/j.apal.2013.05.009. Special issue on Seventh Workshop on Games for Logic and Programming Languages (GaLoP VII).

[11] Bob Coecke, Mehrnoosh Sadrzadeh & Stephen Clark (2010): *Mathematical foundations for a compositional distributional model of meaning*. Available at `https://arxiv.org/abs/1003.4394`.

[12] Antonin Delpeuch (2017): *Autonomization of Monoidal Categories*. doi:10.31219/osf.io/efs3b.

[13] Vedran Dunjko & Hans J Briegel (2018): *Machine learning & artificial intelligence in the quantum domain: a review of recent progress*. *Reports on Progress in Physics* 81(7), p. 074001, doi:10.1088/1361-6633/aab406.

[14] Vedran Dunjko, Jacob M. Taylor & Hans J. Briegel (2017): *Advances in quantum reinforcement learning*. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, doi:10.1109/smc.2017.8122616.

[15] Lawrence Dunn & Jamie Vicary (2019): *Coherence for Frobenius pseudomonoids and the geometry of linear proofs*. *Logical Methods in Computer Science* Volume 15, Issue 3, doi:10.23638/LMCS-15(3:5)2019.

[16] Giovanni de Felice, Konstantinos Meichanetzidis & Alexis Toumi (2019): *Functorial Question Answering*. Available at `https://arxiv.org/abs/1905.07408`.

[17] Giovanni de Felice, Alexis Toumi & Bob Coecke (2020): *DisCoPy: Monoidal Categories in Python*.

[18] Angel J. Gallego & Roman Orus (2017): *Language Design as Information Renormalization*. Available at `https://arxiv.org/abs/1708.01525`.

[19] Michael R Garey & David S Johnson (1983): *Crossing number is NP-complete*. 4, SIAM, pp. 312–316, doi:10.1137/0604033.

[20] Vittorio Giovannetti, Seth Lloyd & Lorenzo Maccone (2008): *Architectures for a quantum random access memory*. *Physical Review A* 78(5), doi:10.1103/physreva.78.052310.

[21] Stefano Gogioso (2016): *A Corpus-based Toy Model for DisCoCat*. *Electronic Proceedings in Theoretical Computer Science* 221, p. 2028, doi:10.4204/eptcs.221.3.

[22] E. Grefenstette & M. Sadrzadeh (2011): *Experimental Support for a Categorical Compositional Distributional Model of Meaning*. In: *The 2014 Conference on Empirical Methods on Natural Language Processing.*, pp. 1394–1404. Available at `https://arxiv.org/abs/1106.4058`.

[23] Vojtch Havlek, Antonio D. Crcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow & Jay M. Gambetta (2019): *Supervised learning with quantum-enhanced feature spaces*. *Nature* 567(7747), p. 209212, doi:10.1038/s41586-019-0980-2.

[24] D. Kartsaklis & M. Sadrzadeh (2013): *Prior disambiguation of word tensors for constructing Sentence vectors*. In: *The 2013 Conference on Empirical Methods on Natural Language Processing.*, ACL, pp. 1590–1601. Available at `https://core.ac.uk/display/102029401`.

[25] D. Kartsaklis & M. Sadrzadeh (2014): *A Study of Entanglement in a Categorical Framework of Natural Language*. In: *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*, doi:10.4204/EPTCS.172.17.

[26] Joachim Lambek (2006): *Pregroups and natural language processing*. *The mathematical intelligencer* 28(2), pp. 41–48. Available at `10.1007/BF02987155`.

[27] Joachim Lambek (2008): *From Word to Sentence: a computational algebraic approach to grammar*. Polimetrica sas.

[28] M. Lewis (2019): *Compositional Hyponymy with Positive Operators*. In: *Proceedings of Recent Advances in Natural Language Processing, Varna, Bulgaria*, pp. 638–647. Available at `10.26615/978-954-452-056-4_075`.

[29] Martha Lewis & Bob Coecke (2016): *Harmonic Grammar in a DisCo Model of Meaning*. Available at `https://arxiv.org/abs/1602.02089`.

[30] Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh & Matthew Purver (2014): *Evaluating Neural Word Representations in Tensor-Based Compositional Settings*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, doi:10.3115/v1/d14-1079.

[31] Vasily Pestun & Yiannis Vlassopoulos (2017): *Tensor network language model*. Available at `https://arxiv.org/abs/1710.10248`.

[32] Anne Preller (2007): *Linear Processing with Pregroups*. *Studia Logica: An International Journal for Symbolic Logic* 87(2/3), pp. 171–197. Available at `http://www.jstor.org/stable/40210807`.

[33] John Preskill (2018): *Quantum Computing in the NISQ era and beyond*. *Quantum* 2, p. 79, doi:10.22331/q-2018-08-06-79.

[34] M. Sadrzadeh, S. Clark & B. Coecke (2013): *The Frobenius anatomy of word meanings I: subject and object relative pronouns*. *Journal of Logic and Computation* 23(6), p. 12931317, doi:10.1093/logcom/ext044.

[35] Mehrnoosh Sadrzadeh, Stephen Clark & Bob Coecke (2014): *The Frobenius anatomy of word meanings II: possessive relative pronouns*. *Journal of Logic and Computation* 26(2), p. 785815, doi:10.1093/logcom/exu027.

[36] Mehrnoosh Sadrzadeh, Dimitri Kartsaklis & Esma Balkir (2018): *Sentence entailment in compositional distributional semantics*. *Annals of Mathematics and Artificial Intelligence* 82(4), pp. 189–218, doi:10.1007/s10472-017-9570-x.

[37] Maria Schuld, Ilya Sinayskiy & Francesco Petruccione (2014): *An introduction to quantum machine learning*. *Contemporary Physics* 56(2), p. 172185, doi:10.1080/00107514.2014.964942.

[38] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *$t|ket\rangle$: A Retargetable Compiler for NISQ Devices*.

[39] Nathan Wiebe, Alex Bocharov, Paul Smolensky, Matthias Troyer & Krysta M Svore (2019): *Quantum Language Processing*. Available at `https://arxiv.org/abs/1902.05162`.

[40] G. Wijnholds & M. Sadrzadeh (2019): *Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings*. In: *Proceedings of the 2019 Conference of the ACL*, Association for Computational Linguistics, pp. 261–271, doi:10.18653/v1/N19-1023.

[41] Peter Wittek (2014): *Quantum Machine Learning*. Academic Press, doi:https://doi.org/10.1016/B978-0-12-800953-6.00018-9.

[42] William Zeng & Bob Coecke (2016): *Quantum Algorithms for Compositional Natural Language Processing*. *Electronic Proceedings in Theoretical Computer Science* 221, p. 6775, doi:10.4204/eptcs.221.8.