A Compositional Approach to Reading Comprehension Tasks Using the DisCoCirc Natural Language Processing Framework.



Tiffany Duneau

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2024

Acknowledgements

In the spirit of DisCoCirc, here's a diagram containing people who have helped bring this thesis into existence. Thank you.



Statement of Authorship

Parts of this thesis are based on the following joint work:

> Tiffany Duneau, Saskia Bruhn, Gabriel Matos, Tuomas Laakkonen, Katerina Saiti, Anna Pearson, Konstantinos Meichanetzidis, and Bob Coecke. Scalable and interpretable quantum natural language processing: an implementation on trapped ions. (Duneau et al., 2024)

Additionally, parts of the work produced for this thesis have been included in:

> Tiffany Duneau. Towards a Comparative Framework for Compositional AI Models. (Duneau, 2025)

Where possible, the sections included here were predominantly my own work, except where further context was necessary, or in the interest of providing more complete results. The following breaks down my contributions to Duneau et al. (2024), and where they can be found within this thesis.

Contributions to the paper My primary contributions to Duneau et al. (2024) consists of much of the code development work required to implement quantum DisCoCirc as a trainable model, as well as certain optimisations to allow more efficient training. The same code, and further developments thereof, was used to generate the other results reported on in this thesis. Additionally, I oversaw the training for the two-directional quantum models, as well as the interpretability of the two- and four-directional models using the Bloch sphere representations. Finally, I also conducted the resource analysis of the datasets.

Sections included in this thesis Chapter 3, Section 3.1 and Section 3.3.1 introduce the *following* dataset from Duneau et al. (2024). The training methodology sections are reproduced in Chapter 5, Section 5.3.2. Section 6.1 of Chapter 6 is mostly taken directly from the paper, with minor modifications to the figures, and the inlining of certain appendix results. Similarly, the interpretability section (except the section on interventions) and associated appendices are reproduced as Section 7.1 in Chapter 7.

Abstract

The DisCoCirc framework for natural language processing allows the construction of compositional models of text, by combining units for individual words together according to the grammatical structure of the text to be read by the model. The compositional nature of a model can give rise to two things: *compositional generalisation* - the ability of a model to generalise outside its training distribution by learning compositional rules underpinning the entire data distribution - and *compositional understandability* - making sense of how the model works by inspecting its modular components in isolation and the processes through which they are combined.

We present a proof-of-concept implementation of a DisCoCirc model on a trapped-ion quantum computer, solving a hand-crafted question answering task. DisCoCirc allows a quantum model to be trained in classical simulation on small instances, while larger instances at test time are evaluated on a quantum computer. This avoids the barren plateau pitfalls of other common QML setups. We find that the accuracy of the models decay by at most 5% when evaluated on the *Test* set: the model compositionally generalises.

Secondly, we consider how well DisCoCirc models can learn different types of compositional behaviour. We compare both quantum circuit based models, as well as classical neural networks under a series of training paradigms, on a dataset derived from one of the bAbI tasks, extended to test a series of aspects of compositionality. Both architectures score within 5% of one another on the productivity and substitutivity tasks, but differ for the systematicity and overgeneralisation tasks. Overall, we find the neural models are more prone to overfitting the *Train* data.

Finally, we inspect the trained models, both by comparing them to manually-constructed perfect compositional models, and by considering how the model components interact with one another, explaining how the models behave and exhibiting the models' compositional understandability.

Contents

0 Introduction **8**

0.1 The gap . . . 9

0.1.1 Understanding the model . . . 9

0.1.2 The model's understanding . . . 10

0.1.3 Neuro-Symbolic AI . . . 11

0.2 Reading comprehension . . . 14

0.3 Outline and contributions . . . 15

I Background **16**

1 Compositionality **17**

1.1 Definitions . . . 18

1.1.1 Atoms and rules (languages) . . . 19

1.1.2 Compositionality . . . 22

1.1.3 Useful compositionality . . . 24

1.2 Aspects of compositionality . . . 26

1.2.1 Systematicity . . . 28

1.2.2 Productivity . . . 29

1.2.3 Substitutivity . . . 31

1.2.4 Overgeneralisation . . . 32

1.2.5 Localism . . . 33

1.3 Required assumptions . . . 35

1.3.1 English is compositional . . . 35

1.3.2 Question answering is compositional . . . 37

2 DisCoCirc **38**
2.1 Notation . . . 38
2.1.1 Monoidal category . . . 38
2.1.2 Rigid . . . 40
2.1.3 Braided . . . 40
2.1.4 Discard . . . 41
2.1.5 Frame expansion . . . 42
2.2 DisCo- formalisms . . . 45
2.2.1 Frameworks . . . 47
2.3 Question answering . . . 52
2.3.1 Scope . . . 53
2.3.2 Implementations . . . 54

II Methodology **57**

3 Tasks and datasets **58**
3.1 Tasks . . . 58
3.1.1 Following . . . 58
3.1.2 bAbI 6 . . . 62
3.2 Analysis and characterisation . . . 63
3.2.1 Compositionality . . . 63
3.2.2 Interpretability . . . 65
3.2.3 Resources . . . 66
3.3 Datasets . . . 67
3.3.1 Following . . . 67
3.3.2 bAbI 6 . . . 73

4 Models **81**
4.1 Quantum . . . 81
4.1.1 Constraints . . . 82
4.1.2 Implementation . . . 84
4.1.3 Future work . . . 84
4.2 Classical . . . 87
4.2.1 Implementation . . . 89
4.2.2 Future work . . . 89

5 Training methodology **91**
5.1 Pipeline . . . 91
5.2 Training . . . 93
5.3 Optimising compositionality . . . 93
5.3.1 Training regimes . . . 93
5.3.2 Following . . . 96
5.3.3 bAbI 6 . . . 97

III Results **100**

6 Compositional generalisation **101**
6.1 Proof-of-concept: following . . . 101
6.1.1 Cross-validation . . . 104
6.1.2 By density . . . 105
6.1.3 H1-1 and *Test (Qujax)* sub-datasets . . . 106
6.2 Aspects of compositionality: bAbI 6 . . . 110
6.2.1 Overgeneralisation . . . 113
6.2.2 Training for compositionality . . . 115
6.2.3 Cross-validation . . . 119

7 Compositional understandability **122**
7.1 Following datasets . . . 123
7.1.1 Hand-crafted Clifford models . . . 124
7.1.2 Trained model: two-directional dataset . . . 130
7.1.3 Trained model: four-directional dataset . . . 136
7.1.4 Discussion . . . 143
7.2 bAbI 6 . . . 145
7.2.1 Productivity: quantum model . . . 146
7.2.2 Discussion . . . 154

8 Conclusions **156**
8.1 Discussion . . . 157
8.2 Future work . . . 158
8.2.1 Beyond . . . 162

References **163**



CHAPTER 0

Introduction

In recent years, the stage for Natural Language Processing (NLP) and Artificial Intelligence (AI) more broadly has seen a huge boom following the advent of a new paradigm for constructing models. Large Language Models (LLMs) such as GPT-4, (OpenAI et al., 2024) and Llama (Touvron et al., 2023) have taken the world by storm and enjoy widespread applications as chat-bots, virtual assistants, code writers and more.

The success of these models lies in their remarkable ability to generate human-like text in response to a wide variety of questions and tasks (Bubeck et al., 2023). By contrast, our capacity to understand why such models give certain answers, and the content of these answers on certain types of question are not always quite as remarkable (Arkoudas, 2023).

DisCoCirc (Coecke, 2020) is a framework that intends to address such issues, providing a method for building models that are easier to for humans to interpret. We address reading comprehension tasks in this work, as a way to probe the internal representations, or 'meanings' a model learns to give to pieces of text.

Underlying this whole thesis, will be a consideration of compositionality, both as a tool that enables the qualities of the DisCoCirc framework, and also as an assumption regarding the kind of tasks we want to consider. While a lengthy debate surrounds the idea that compositionality underpins all of human thought and language use (see eg. Fodor and Pylyshyn (1988); Smolensky (1991) and subsequent replies), it is more difficult to deny the importance of compositional phenomena underpinning a wide variety of tasks. Formal languages are typically specified compositionally, such that reasoning with and evaluating their expressions becomes a compositional task, while in the visual domain, a compositional interpretation allows for more sophisticated relationships between images. Putting

aside the question of whether a compositional model is the *most* correct model we can find to account for a given task, in this thesis, we examine some arguments as to why a compositional model may nevertheless be desirable, from both a generalisation perspective, as well as by allowing for a better understanding of how the model behaves.

Before fully stepping into compositionality, however, let us take a closer look at the state of current models.

0.1 The gap

LLMs are an extension and scaling up of the transformer architecture introduced in Vaswani et al. (2017). By allowing the models to have billions of parameters, and training in an unsupervised fashion on corpora scraped from the internet, these models are able to capture increasingly complex functions. As the natural endpoint of the empirical scaling law for transformers (Kaplan et al., 2020), these models exemplify the idea that 'bigger is better'. And indeed, this has largely continued to be the case (though see McKenzie et al. (2024) for exceptions): bigger models are able to handle and synthesise more data, which in turn allows them to model more complex functions of that data. When trained in the right way, this function increasingly looks like meaning.

We characterise two major failings of LLMs in terms of understanding: both in the way that humans can understand why the model acts as it does, as well as the way the model can be seen to understand (or not) the content it generates and responds to.

0.1.1 Understanding the model

The major concern these models bring is their inherent obscurity: even at small scale, it is difficult to interpret what is going on in a transformer. When scaled to LLMs sizes, this task becomes even harder. It is however incredibly important if we are to trust LLMs to inform high stakes decisions, such as advising a patient on their diagnosis or treatment options, or making financial decisions (Rudin, 2019; Vilone and Longo, 2021; Ferdaus et al., 2024). In cases where our lives are at stake, we like being able to understand why decisions are made. If things go wrong it is important to understand where and why so that we can learn and improve.

0.1.2 The model's understanding

Another related class of limitations exhibited by LLMs is a lack of what we might consider true understanding. Though incredibly capable at generating sentences that look right on the surface, in many cases the models are prone to make glaring mistakes when probed further about their responses (Arkoudas, 2023; Jiang et al., 2024).

Understanding is a notoriously difficult concept to both pin down and assess, however the most prominent and successful method we have to test this is to ask questions. These questions are designed to probe at supposed internal representations, in order to check that they are suitably aligned with those of the asker. Indeed this is how we can ensure that communication is possible at all - we each have our own internal understanding of what words mean. Assuming these are broadly aligned, when Alice expresses some internal concept in a language she shares with Bob, a successful interaction results in Bob receiving this internal notion. They can then engage in a series of back and forth questions about the concept in an attempt to prove to each other that they both understand what they're talking about. Of course we could construct pathological examples where Alice and Bob each have sufficient words that do not correspond to the other's that they are able to maintain two parallel conversations: the one Alice thinks they are having and the one Bob thinks they are having. In most cases, such a phenomenon will be noticed quite quickly and the discrepancy addressed.

Compositional reasoning

A notable sub-branch of understanding deals with compositional reasoning. Certain tasks can be solved via the application and chaining together of rules that manipulate some simple base elements. Logic-based tasks are a paradigmatic example - from an initial set of premises and selection of axioms, everything else can be derived as combinations of those elements. This sort of reasoning is known as compositional (more on this in Chapter 1), and crucially, is a type of reasoning that LLMs (and Transformers more generally) have historically struggled with (Mirzadeh et al., 2024; Dziri et al., 2023).

Meanwhile, there is significant literature that explores such reasoning in the symbolic AI community, with great success. Theorem provers and assistants (Barendregt and Geuvers, 2001), SAT solvers (Alouneh et al., 2019), and knowledge graphs (Yahya et al., 2012) are some of the major, successful outputs of this line of research. Their inherent reliance on logical formulae acts as a double-edged

sword however, as while ensuring correctness and making such methods easy to interpret, it also makes these techniques hard to apply in realistic situations where the appropriate logical formulae may not be obvious or easy to generate. Indeed, it is very easy to come up with paradoxical sentences ('*This sentence is not true.*') that such models may not be able to handle.

0.1.3 Neuro-Symbolic AI

Recently, there has been work that attempts to combine these seemingly orthogonal approaches by giving LLMs access to symbolic oracles (Davis and Aaronson, 2024). In this way, an LLM can be used to broaden the scope of the symbolic methods, by providing an automated translation of problems into the correct format, whilst offloading the more intensive reasoning aspects that they struggle with.

Another paradigm involves making the base representations of a symbolic approach learnable, for example by creating vector representations of entities and relations in knowledge bases (Huang et al., 2019).

Distributional Compositional Models

The Distributional Compositional (DisCo-) frameworks take a similar approach, in the context of NLP. The structure of the model is hence guided by formal rules (namely grammar) - the compositional part, whilst the atomic meanings these rules deal with and combine are learnable - the distributional part. Indeed, in Coecke et al. (2018) a class of question answering problems in DisCoCat is framed in terms of knowledge graph queries. Figure 1 visualises the DisCo- approaches, and their relation to the surrounding AI and NLP work.

Though not as mature and effective (yet) as Transformer-based models, the DisCo- frameworks provide models that preserve the interpretability benefits of symbolic approaches, most notably due to their compositional nature. As explored in Chapter 1, and indeed the rest of this work, this compositionality can also make them effective at solving tasks requiring compositional reasoning, provided the same rules are used to solve the task as to build the model.

Taking a step back, distributional semantics suggests that meanings of words should be modelled by the meanings of their surrounding words, and this can also be extended to entire pieces of text and their constituent words. Approaches like Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014a) exemplify

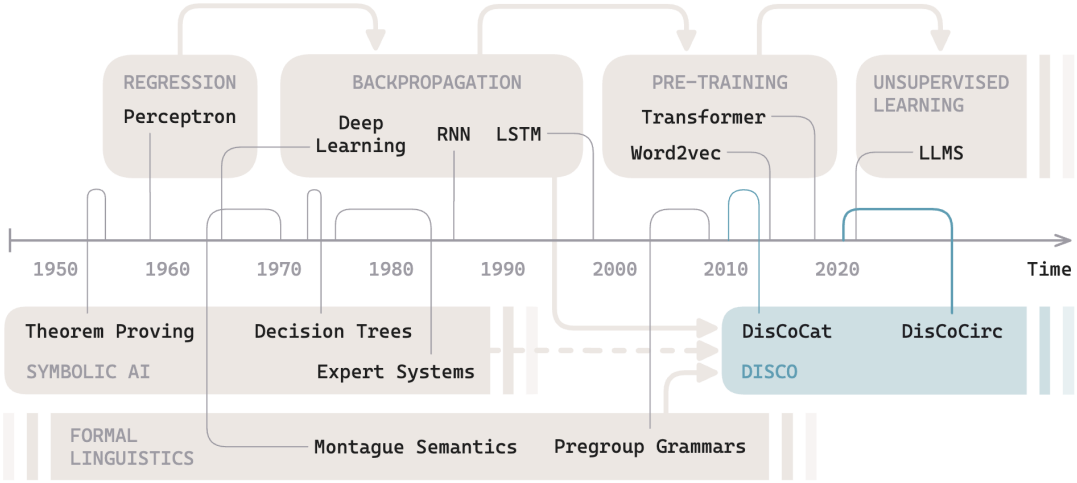


Figure 1: A timeline sketch of certain key developments in the history of AI and machine learning that are relevant to this work. The various approaches are grouped roughly according to the predominant paradigm underpinning their development, or the general subfield to which they belong.

this by computing vector embeddings for words capturing the similarity of the contexts they appear in.

Meanwhile compositional semantics considers how meaning can arise from the grammatical form of a sentence, using syntactic operations to derive the meaning of an entire sentence from some atomic word meanings. Montague semantics (Montague, 1970, 1973) is a prototypical example, in which sentences belonging to a fragment of English are converted into logical formulae via a deterministic, compositional function.

The DisCo- approach to modelling meaning is then a method for constructing embeddings for natural language inputs, by allowing the atoms to take distributional semantics that can be learnt either in-task or pre-trained, and subsequently combining these into a single representation for an entire sentence (DisCoCat, Coecke et al. (2010)) or an entire text (DisCoCirc, Coecke (2020)) using predetermined rules dependent on the syntax of the given sentence.

The DisCo- based frameworks are still relatively new, and are only beginning to see some larger scale experiments performed. These are generally targeted towards a specific task, though in many cases only certain aspects of the overall model are trained or under consideration. We group the work thematically.

Grammar Anaphora and ellipsis resolution are looked into in Wijnholds and Sadrzadeh (2019a,b), making use of a more sophisticated grammar to deal with the required copying of words and phrases across the sentence. Functional words like relative pronouns have also been investigated in Sadrzadeh et al. (2013, 2016) and given syntactic representations, reflecting the fact that the the word meanings cannot be accurately inferred from their typical context, since they tend to occur in almost all contexts.

Logic Various methods of computing sentence-level entailment have been investigated. Bankova et al. (2016); Balkir et al. (2018) explores using a density matrix based formalism to encode meanings; Lewis (2019) derives encodings using a combination of pre-trained encodings and WordNet (Fellbaum, 1998) to construct encodings hierarchically; and Cuevas et al. (2020) investigates using positive semi-definite matrices. These approaches remain restricted to sentences that share the same grammatical structure, and effectively only test how word-level entailment spreads to the sentence level in upward monotone sentences. A similar construction has also been used to model ambiguity in Piedeleu et al. (2015).

Some work is also being done towards explicit modelling of logical connectives. Lewis (2020) explores how to model negation compatibly with a choice of entailment measure, whilst Rodatz et al. (2021); Shaikh et al. (2021) considers conversational (rather than logical) negation. Conjunctions have also been treated for single sentences in Kartsaklis (2016), and for DisCoCirc specifically in Duneau (2021a).

Languages Much of the initial work has focused on the English language. Looking beyond English, Waseem et al. (2022) uses DisCoCirc to analyse Urdu. Balthasar Mansky et al. (2023) applies DisCoCat to Chinese, and earlier work analyses many further languages using pregroup grammars (Lambek, 2006, 2008), a grammar that is often taken as a basis for the DisCo- frameworks.

Question Answering The paper de Felice et al. (2020) investigates using the DisCoCat model to learn how to answer a question, by framing it as a gap-filling exercise for the sentence, while Meichanetzidis et al. (2020) implements a toy quantum model that determines whether a given sentence is true or not. Lorenz et al. (2021) extends this work, training quantum DisCoCat models to solve binary classification tasks. Further, Kartsaklis et al. (2021) creates a toolkit for

implementing DisCoCat on both classical and quantum hardware, highlighting the progression towards larger scale models within DisCoCat, and recently DisCoCirc.

On a more theoretical side, Laakkonen et al. (2024) investigates question answering for DisCoCirc, while Zeng and Coecke (2016); Correia et al. (2022) suggest algorithms for working with DisCoCat.

This work builds upon these approaches and presents the first proof-of-concept implementation of question answering with DisCoCirc.

0.2 Reading comprehension

One of the main goals to aim for in constructing a language model is *understanding* - ultimately we would like our models to respond to the text they interact with in a way that displays an understanding of the relevant concepts. As highlighted in Section 0.1.2, however, this is not particularly easy. Here we briefly consider how such understanding may be evaluated.

In order to test whether an agent understands human language, we must first find some suitable interpretation of what *understanding* even is. Unfortunately, this is much easier said than done, though philosophers do tend to agree that understanding consists in some way of finding links between objects in a given domain. The exact nature of these objects varies (Kim, 1994), but the main theme remains, along with a notion that understanding goes beyond knowledge in the sense that it concerns counterfactual claims, not just claims about the state of the actual world. In particular, if we want to claim to understand why the glass broke when it was dropped onto the floor, we should also have ideas about what would happen if the glass were dropped onto a pillow instead. Further, the framing of understanding as *connections* tends to discount brute-force type methods like Block (1981)'s blockhead that merely regurgitate pre-learnt answers to whatever question is posed.

Given this, constructing a way to test whether an agent has understood a passage of text should then be focused on probing any connections the agent might have made between the words and sentences involved, perhaps to other sentences or ideas. Reading comprehension is a task that purports to fill this role: given a piece of text, the reader is posed questions about it. At the most basic level, this can probe sentence-level connections, determining whether the reader has found the proper relation between the words - what is happening, who

is doing it and who it is happening to, for example. More complex questions might then require connections to facts or events not featuring in the text directly, or making higher-level connections between sentences in the text, potentially involving inference.

In this work, we focus on basic, binary question answering tasks using synthetic datasets, as a first step towards solving more general reading comprehension tasks.

0.3 Outline and contributions

The work is structured as follows: Part I sets to fill in the relevant theoretical background underlying this work. We start with an exploration of compositionality in Chapter 1. The key notions of *compositional generalisation*, and *compositional understanding* are introduced, as well as methods for quantifying how compositional a model is in a practical setting. Additionally, compositionality as applied to language and machine learning models is presented within a categorical framework. The rest of the work is concerned with testing and measuring the concepts laid out here, for the specific case of DisCoCirc models.

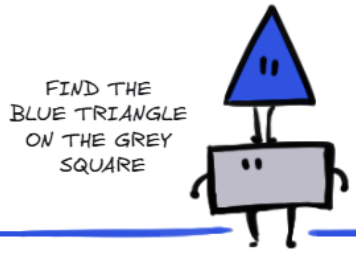
With this context in mind, Chapter 2 presents the DisCo-models and details how they may be used for the question-answering type tasks considered on a theoretical level. This is also where a brief overview of the category theory underpinning the diagrammatic formalism of DisCoCirc is provided. From this point on, familiarity with monoidal diagrams will be assumed.

Part II is then concerned with the particular implementation details: Chapter 3 establishes the tasks the models are to be evaluated on, and provides details on dataset construction. The specific implementations of the DisCoCirc framework as a quantum circuit model and neural-network based model are detailed in Chapter 4, while the particulars of actually training these models are documented in Chapter 5.

Finally, Part III details the results obtained, following the notions of compositional generalisation and understandability introduced in Chapter 1. We evaluate the models' capacities for compositional generalisation in Chapter 6, via both an initial proof-of-concept work with the quantum model, as well as a more detailed study over four aspects of compositionality. In Chapter 7, a selection of these trained models are decomposed and interpreted, showing how their compositional structure leads to compositonal understandability. Finally Chapter 8 summarises the results and indicates a list of directions for future work.

Part I

Background



CHAPTER 1

Compositionality

The concept of compositionality provides the backdrop that situates this thesis. In this chapter, we define what we mean when we call a task, language, or model compositional, and identify two applications that make such a property interesting: compositional generalisation and compositional understandability, as well as five sub-properties or aspects of compositionality that we will later use to evaluate the success of our models. The work in this chapter provides an initial pass at formalising the concepts involved; these are further developed in Duneau (2025)[1], alongside examples.

Informally, we can consider a process to be compositional if it requires only two ingredients in order to accomplish its goal: (i) basic atoms, and (ii) a strategy for combining these atoms. Formalising what compositionality means for a given goal is then intrinsically tied to a specification of what kind of things are atoms, what kind of rules are available when combining them, and what kind of operations are allowed on these inputs as part of the processing.

For the particular process of assigning meaning to sentences of a natural language (say, English), there has been much debate surrounding whether it is compositional; if so, how to formalise such a thing, and even whether the notion of compositionality has any empirical content at all (Partee, 1984a; Pelletier, 1994; Szabó, 2012).

Meanwhile, for other goals, category theory has proved an ideal tool for formalising and reasoning about such a notion. Indeed, the presentation of a monoidal category echoes the above dichotomy: it is inherently a story about

[1]Note that the paper calls compositional understandability (as defined in this thesis) compositional *interpretability*.

atoms (objects and morphisms), and what it means to put them together. More generally, we can consider categories with extra structure as a way to formalise what the rules in question are. As for the allowed operations: the specification of a structure-preserving functor between categories seems an ideal way to capture the sort of compositional-friendly operations we might want our process to consist of.

In this thesis, we shall approach the problem from a slightly different angle: rather than trying to find a compositional account of meaning in general, we will be interested in learning approximations to this process - natural language processing models - that are compositional in a useful way, for a restricted fragment of language that avoids the pitfalls set out in Partee (1984a).

In order to do so, this chapter will lay the theoretical groundwork underpinning the experimental work in the rest of this thesis: in Section 1.1, we will consider compositionality to be a property of a process, taken *relative to a choice of rules* (the syntax) which may or not apply to the following:

(1) The meaning of a natural language, such as English.

(2) The internal structure of a model (*syntactic*).

(3) The function a model implements (*semantic*).

We distinguish situations (2) and (3), introducing the respective notions of *syntactic* and *semantic* compositionality to describe them. Ultimately, our aim is to ensure that the syntax relative to which a particular model and natural language task are compositional, are in fact one and the same: this is what makes the compositionality of the model useful.

Section 1.2 lays out five aspects of compositionality that can be used in two ways: first as a way to ensure that the compositional rules are appropriate, and second as a way to test empirically whether a model is compositional.

Finally, in Section 1.3, we elaborate on which language fragments might be considered safely compositional, and consider in more detail how this extends to NLP tasks like question answering.

1.1 Definitions

We want to consider compositionality as a property of a process, which tells us something interesting about how its outputs are related to its inputs. We will

primarily be interested in *functors*. In the context of ML and solving a particular task, we will also call them *models*.

Definition 1. *A functor, written $f : \mathcal{A} \to \mathcal{B}$, is a mapping between categories $\mathcal{A}$ and $\mathcal{B}$, that preserves identities and the composition of morphisms. f sends each object in $\mathcal{A}$ to an object in $\mathcal{B}$, and each morphism $a : A \to A'$ in A to a morphism in $\mathcal{B}$ such that $f(a) : f(A) \to f(A')$.*

A function in the usual sense (as a mapping between sets) can be expressed as a functor between discrete categories, in which the objects are indexed by members of a set, and the only morphisms are identities. In this work, the domain will usually consist of sentences from a specified language, L, which we package as a category where each sentence is an isolated morphism.

1.1.1 Atoms and rules (languages)

Formally, a language L is a set of sentences, Σ_L, each composed of atoms $a \in \mathcal{A}$. However Σ_L may not contain all possible combinations of the atoms. One way of specifying a language is by an inclusion rule, for example as a set comprehension: $\{a^n b^n : n \in \mathbb{N}\}$. In this case, $\mathcal{A}$ would be the set $\{a, b\}$. Thanks to Chomsky (1965), we can also present a language via a generative grammar. Here, our goal is to translate such grammars into a categorical setting.

Informally, a generative grammar is a set of production rules $\mathcal{R}$ that specify how to construct all and only sentences of Σ_L. Each rule can be understood as a way to replace a non-empty list of symbols with a (possibly empty) list of new symbols. We can write $R : s_0 \cdot ... \cdot s_n \to t_0 \cdot ... \cdot t_n$ for the rule that allows the string of symbols $s_0 \cdot ... \cdot s_n$ to be replaced with the string $t_1 \cdot ... \cdot t_n$. Valid sentences are taken to be those obtained by iteratively applying rules to a starting symbol t_*, until all remaining symbols are atoms $a \in \mathcal{A}$. More formally:

Definition 2. *A generative grammar is a tuple $(\mathcal{A}, \mathcal{T}, \mathcal{R}, t_*)$ where:*

- *$\mathcal{A}$ is a finite collection of atoms.*
- *$\mathcal{T}$ is a finite collection of types.*
- *$\mathcal{R}$ is a finite collection of rules of the form*

$$R_{base} : b_0 \cdots b_n \to a$$
$$R_{complex} : s_0 \cdots s_n \to t_0 \cdots t_m$$

where $a \in \mathcal{A}$, $b_i, s_i, t_j \in \mathcal{T}$ *and* $n \geq 1, m \geq 0$.

- $t_* \in \mathcal{T}$ *is a designated 'starting' type.*

Note that there could be multiple rules that apply to a given type $t \in \mathcal{T}$. Without loss of generality, we can enforce the $R_{base}/R_{complex}$ split by introducing intermediate symbols as necessary to defer the introduction of atoms into a separate rule. Notice that $\mathcal{A}$, $\mathcal{T}$ and $\mathcal{R}$ are *finite*, while Σ_L may be infinite. This is important.

Typically, a generative grammar is paired with a parser, which, given a sequence of atoms, tells us how it can be decomposed as a sequence of grammatical rules, and corresponding intermediate type sequences. This sequence of rules is known as a syntax derivation. For many practical purposes, this is assumed to be tree-shaped for tractability (as we shall see later, context-free grammars are this shape), however it is most generically a directed graph, or indeed a diagram in a monoidal category.

Derivations whose final type is t_* are called sentences, which we say the parser recognises. Conversely, every sequence of atoms that can be obtained from successive applications of rules to the starting type t_* is a sentence that the grammar generates. We say that a grammar generates (and the associated parser recognises) a language L when the set of sentences it generates (respectively, recognises) is exactly Σ_L. There may be multiple grammars that generate the same language.

Syntax

We can very naturally translate generative grammars into monoidal signatures, from which we can derive a free monoidal category.

Recall that a monoidal signature is the following data:

- A finite set of base objects.
- A monoidal product $\otimes$.
- A finite set of base morphisms, with source and targets taken as iterated monoidal products of the base types.

Which can be extended to a monoidal category by allowing the objects to be defined as the free monoidal compositions of the base objects under $\otimes$, and the morphisms to be the freely (type-checking) compositions of base morphisms,

subject to the usual coherence conditions for monoidal categories. We call each particular composition of morphisms a diagram, as it can be drawn as such using the graphical calculus of monoidal categories, as presented in e.g. Selinger (2011).

Definition 3. *A generative grammar* $(\mathcal{A}, \mathcal{T}, \mathcal{R}, t_*)$ *gives rise to a grammatical signature* $\mathcal{G}$*:*

- *Base objects are indexed by the types* $t \in \mathcal{T}$ *along with a special unit type* I*.*
- *The monoidal product is written* $\cdot$*, representing concatenation.*
- *Base morphisms are indexed by the rules, with type signature given by* $R : t_0 \cdots t_n \to s_0 \cdots s_n$ *for each rule* $R_{complex} \in \mathcal{R}$*. The terminal rules* R_{base} *are called states, and have special signature* $R : \mathsf{I} \to b_0 \cdots b_n$*. Notice that the type signature for the morphisms is the inverse of the rules. This arises from a difference between parsing and generating - ultimately we want the diagram to represent a parsing, while the grammar is usually used for generation.*

We call the associated monoidal category the syntax category $S_{\mathcal{G}}$ of the grammar. Notice that not all of the objects of $S_{\mathcal{G}}$ will represent valid sentences. An important aspect that is left undetermined here is the kind of structures allowed or imposed on the types and morphisms - various options are explored in Duneau (2025). These choices will ultimately determine the complexity of the grammar, as well as its practical use for a particular task.

For convenience, define $\Sigma_{\mathcal{L}}$ to be the category whose objects are S_s and S_t for each sentence $S \in \Sigma_L$, with morphisms $S : S_s \to S_t$, for each sentence (and additional identity morphisms). We can then define a syntax functor:

Definition 4. *A syntax for a language* L *is a functor* $e : \Sigma_{\mathcal{L}} \to S_{\mathcal{G}}$*, assigning a diagram of* $S_{\mathcal{G}}$ *to each sentence of* L*, according to its syntax derivation. We write* $\mathsf{D}_{\mathcal{G}}(L)$ *to denote the diagrams in* $S_{\mathcal{G}}$ *selected by* e *- these are the diagrams corresponding to sentences of* L*. Equivalently, this is the set of diagrams in* $S_{\mathcal{G}}$ *with codomain* t_**.*

Semantics

Languages are also generally equipped with an interpretation, which allows us to assign meaning to the sentences involved.

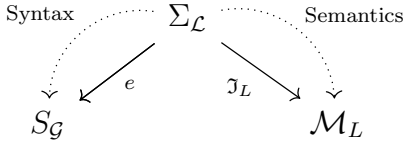
Definition 5. *An interpretation of a language L is a functor $\mathfrak{I}_L : \Sigma_{\mathcal{L}} \to \mathcal{M}_L$, that sends sentences into a meaning category $\mathcal{M}_L$. We call this the semantics of L.*

Importantly, we impose as little structure as possible on the interpretation $\mathfrak{I}_L$ - $\mathcal{M}_L$ is not required to have any particular structure. Typically, the models we will encounter implement an interpretation over the language of a given task, where $\mathcal{M}_L$ contains the space of possible responses.

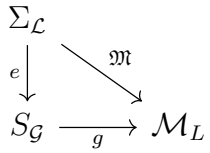
There are two interpretations that we take special notice of. We write $\nabla : \Sigma_{\mathcal{L}} \to \mathcal{M}_L$ for the interpretation that assigns the correct task labels to each input in Σ_L. The second interprets a language for *us* by computing the *meaning* of L, written $\mathfrak{I}^*_L : \Sigma_{\mathcal{L}} \to \mathcal{M}^*_L$. Although a language may not have a unique interpretation $\mathfrak{I}^*_L$, or choice of meaning category $\mathcal{M}^*$, here we will simply fix one such interpretation arbitrarily and ignore the others. Notice that the target meaning category for $\mathfrak{I}^*_L$ and ∇ will typically be different - $\mathcal{M}^*_L$ is something that captures human-understandable meanings, while $\mathcal{M}_L$ need only contain the possible answers for the task.

1.1.2 Compositionality

We can now formalise our two notions of compositionality. The naming distinction effectively reflects which 'branch' of the language data we use to analyse the model on its way to $\mathcal{M}_L$:



Definition 6 (Syntactic compositionality)**.** *A model $\mathfrak{M} : \Sigma_{\mathcal{L}} \to \mathcal{M}_L$ is syntactically compositional with respect to a syntax $e : \Sigma_{\mathcal{L}} \to S_{\mathcal{G}}$ when the following diagram commutes, for an explicitly specified functor $g : S_{\mathcal{G}} \to \mathcal{M}_L$, that preserves the chosen structure of $S_{\mathcal{G}}$:*



In other words $\mathfrak{M}$ is syntactically compositional if there is some composite functor $e; g$ that $\mathfrak{M}$ agrees with for all sentences in Σ_L, such that analysing $\mathfrak{M}$ *just is* analysing e and g. This is quite a strong property however. Recall that we have not restricted the way that $\mathfrak{M}$ is implemented - and in particular it need not

implement this function with any reference to $S_{\mathcal{G}}$ at all. This includes painstaking human annotations, neural networks and lookup tables, as well as implementations that factor through $S_{\mathcal{G}}$ explicitly. Finding a specific g and proving that $\mathfrak{M} = e; g$ may hence be very difficult.

In some cases, however we may be able to prove a weaker property, where g need not be specified explicitly:

Definition 7 (Semantic compositionality)**.** *A model $\mathfrak{M} : \Sigma_{\mathcal{L}} \to \mathcal{M}_L$ is semantically compositional with respect to a syntax $e : \Sigma_{\mathcal{L}} \to S_{\mathcal{G}}$ when there exists a (possibly not explicitly specified) functor $g : S_{\mathcal{G}} \to \mathcal{M}_L$ that decomposes $\mathfrak{M}$ via the syntax, such that $\mathfrak{M} = e; g$.*

This distinction is particularly relevant when we are interested in explaining or interpreting what a functor is doing, when its implementation is obscure (as is the case with neural networks, for example). Rather than treating the function as a black box, syntactic compositionality provides a way of breaking $\mathfrak{M}$ into a series of smaller, and hopefully easier to understand, black boxes, whilst semantic compositionality can allow us to reason about the model's behaviour directly, without needing to open the black box at all.

We will often also require syntactic compositionality relative to a *specific* g, arising as the decomposition of some task labels $\nabla = e_{\nabla}; g$, for example when trying to build a function that approximates labels ∇ that is inconvenient to compute via g. It is important that g preserves the chosen structure of $S_{\mathcal{G}}$ (whether this is merely monoidal, or something more involved such as a rigid or braided category), as this forces it to behave nicely, and prevents the claim from becoming vacuous in some ways. In particular this precludes g from being an arbitrary dictionary over sentences as indexed by their grammatical types, since it must also be defined in a consistent way on all intermediate sub-sentences and atoms occurring in $S_{\mathcal{G}}$. It could still be vacuous, for example if g sends everything to the same object in $\mathcal{M}_L$. Nevertheless, we have defined syntactic compositionality relative to a choice of syntax - if the syntax itself has no interesting structure, being compositional relative to that structure is not an interesting property after all. In Section 1.2 we consider some properties that can help the interpretation and syntax remain interesting, and indeed take these as defining aspects of compositionality, which we further extend into tests that can be used to investigate semantic compositionality.

When we discuss whether a language is compositional, we are really discussing whether its meaning $\mathfrak{I}_L^*$ is compositional with respect to the language's syntax (and analogously for a task with labels ∇):

Definition 8 (Compositional language)**.** *A language L is called compositional when its meaning $\mathfrak{I}_L^*$ is semantically compositional relative to $S_\mathcal{G}$.*

We do not distinguish between semantic and syntactic compositionality at the language or task level as we are not concerned with the specific implementations. It is sufficient that there exists a suitable implementation, such that a syntactically compositional model of $\mathfrak{I}_L^*$ (respectively ∇) can be found. On the other hand, we *do* care about the specific implementation when it comes to providing an actual model to solve a given NLP task, and thus must distinguish syntactic from semantic compositionality of models.

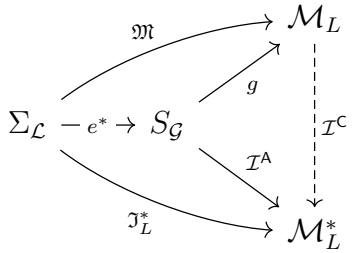
1.1.3 Useful compositionality

As we have seen so far, constructing a syntax that makes a model compositional (either syntactically or semantically) doesn't really guarantee that this compositionality is practically helpful, most notably because the syntax that makes the model compositional might not have the 'right kind' of rules. In many cases, however, we may already have access to some suitable rules, which we package as syntax-interpretation pairs that are compositional relative to this syntax. Two of these in particular are of interest: the meaning $(e^* : \Sigma_\mathcal{L} \to S_\mathcal{G}, \mathfrak{I}_L^* : \Sigma_\mathcal{L} \to \mathcal{M}_L^*)$ and the labels $(e_\nabla : \Sigma_\mathcal{L} \to S_\mathcal{G}, \nabla : \Sigma_\mathcal{L} \to \mathcal{M}_L)$.

Definition 9 (Compositional understandability)**.** *A model $\mathfrak{M} : \Sigma_\mathcal{L} \to \mathcal{M}_L$ is compositionally understandable when:*

- *$\mathfrak{M}$ is syntactically compositional relative to e^*, and*
- *$\mathfrak{I}_L^*$ is syntactically compositional relative to e^*.*

This is related to the compositional interpretability discussed in Tull et al. (2024): the syntactic compositionality of the model guarantees that $\mathfrak{M}$ is a compositional model in the sense required. We can hence derive an abstract interpretation of g (which extends to $\mathfrak{M}$ via the syntactic decomposition $m = e; g$) from $\mathfrak{I}_L^*$ via $S_\mathcal{G}$:



The concrete interpretation $\mathcal{I}^{\mathsf{C}}$ is shown with a dashed line to indicate where it approximately fits in (if it exists),[2] however we will primarily be focusing on the abstract interpretation. Notice that *syntactic* compositionality is required in addition to merely semantic compositionality, as the abstract interpretation only allows us to assign meanings to g (and the elements of $S_{\mathcal{G}}$). Without this condition, such an interpretation might be considered a post-hoc explanation of the model, remaining causally disconnected from the actual implementation of $\mathfrak{M}$: recall that $\mathfrak{M}$ could be anything at all, so long as the answers look right. Such an explanation intuitively feels lacking - after all, $\mathfrak{M}$ could well be implemented as a sentence dictionary, in which case the compositional interpretation has no real bearing on why $\mathfrak{M}$ behaves as it does. To avoid delving too deeply into the philosophy of explanatory theories, we shall only attempt to explain syntactically compositional models where $\mathfrak{M} = e; g$ is what is actually implemented.

That is not to say that semantic compositionality alone is to be discounted - on the contrary, it underpins another significant property.

Definition 10 (Compositional generalisation)**.** *A model $\mathfrak{M} : \Sigma_{\mathcal{L}} \to \mathcal{M}$ compositionally generalises over a syntax $e_\nabla : \Sigma_{\mathcal{L}} \to S_{\mathcal{G}}$ and sentences $\Sigma \subset \Sigma_L$ when:*

- *∇ is semantically compositional relative to e_∇.*
- *If the restrictions $\mathfrak{M}|_\Sigma = \nabla|_\Sigma$, then $\mathfrak{M}|_{\Sigma^*} = \nabla|_{\Sigma^*}$ for the compositional closure Σ^* of Σ with respect to e_∇.*

To see why this is helpful, we consider the restricted signature $\mathcal{G}|_\Sigma$ (with respect to a syntax $e : \Sigma_{\mathcal{L}} \to S_{\mathcal{G}}$, which contains items of $\mathcal{G}$ if and only if they occur in the image of $e|_\Sigma$.

Definition 11. *The compositional closure of a set of sentences $\Sigma \subset \Sigma_L$ with respect to a syntax e is the set of sentences $\Sigma^* \subseteq \Sigma_L$ whose parsing according to e exists in $S_{\mathcal{G}|_\Sigma}$.*

[2]Technically $\mathcal{I}^{\mathsf{C}}$ doesn't go from $\mathcal{M}_L$ but a modified version where we index the objects according to their type in $S_{\mathcal{G}}$.

Thus, for a judicious choice of Σ, the compositional closure just is the language: $\Sigma^* = \Sigma_L$. So, if $\mathfrak{M}$ compositionally generalises over such a set Σ, $\mathfrak{M} = \nabla$, and $\mathfrak{M}$ is also semantically compositional with respect to e_∇. Notice that the grounding clause establishing agreement over Σ is required to ensure that the model is indeed implementing the correct functor - abiding by the same compositional rules as ∇ does not by itself enforce that the atoms are assigned the same base values, which is ultimately what determines the values of sentences.

In Section 1.2, we present a series of aspects of compositionality and associated tests which can allow us to explore the extent to which a model compositionally generalises. Ultimately, compositional generalisation is the property that we look for in models so that we can remain confident in their correctness on the unseen *Test* distribution. Semantic compositionality can provide an pathway to proving this beyond an empirical test, provided that $\mathfrak{M}$ is semantically compositional relative to the same syntax as ∇. In such a case, all that must be established is the grounding clause $\mathfrak{M}|_\Sigma = \nabla|_\Sigma$ over the training set.

1.2 Aspects of compositionality

Establishing whether a model is semantically compositional is difficult without providing a syntactically compositional decomposition, however compositional generalisation allows us to use the syntactic compositionality of *one* model to prove that of another. For the sort of tasks where a compositional approach might be prised, we will typically have independent grounds to believe that the task labels ∇ are semantically compositional relative to some syntax e_∇. A desirable model $\mathfrak{M}$ is then one that both agrees with ∇ on the training set, and moreover is semantically compositional with respect to the same syntax, as this will guarantee that $\mathfrak{M}$ approximates ∇.

Following Hupkes et al. (2020), we identify certain properties a syntax may exhibit - aspects of compositionality - which provide a way to select a training set and corresponding compositional closure for measuring compositional generalisation. These tests are called aspects of compositionality, as they are properties we typically require of a syntax in order to consider functions that factor through it compositional. Recall that very few requirements were placed on the kinds of generators allowed to build the syntax, beyond requiring them to be finite. Though this makes the formalism sufficiently expressive to capture any formal grammar, many of these will be degenerate in some way. The properties from which the tests

are derived can then be viewed as ways to ensure being compositional relative to that syntax is not vacuous. In order to define these tests, we will make use of a utility function acc_A that computes the accuracy of a model over a subset A of Σ_L (as judged by ∇). This is of concern because we want to build models that are compositional relative to the *same* syntax e_∇ as ∇.

We define the compositionality factor of a model, which compares its accuracy on two datasets, A and B. From this, we can derive a compositionality score, which adjusts the model's accuracy over the dataset A according to how well it generalises to the (generally unseen) dataset B. We can view these metrics as measures of compositional generalisation. We define this for a binary task, with baseline accuracy 0.5:

$$\textsc{cFact}(A, B) = \mathsf{max}(0, \mathsf{acc}_\mathrm{A}(\mathfrak{M}) - \mathsf{acc}_\mathrm{B}(\mathfrak{M})) \tag{CF}$$

$$\textsc{cScore}(A, B) = (1 - 2 \cdot \textsc{cFact}(A, B)) \cdot \mathsf{acc}_\mathrm{A}(\mathfrak{M}) \tag{CS}$$

We visualise this function in Figure 1.1. We can also obtain a graded notion of compositionality, ε-compositionality, when a model satisfies:

$$\textsc{cFact}(A, B) \leq \varepsilon \tag{ε-COMP}$$

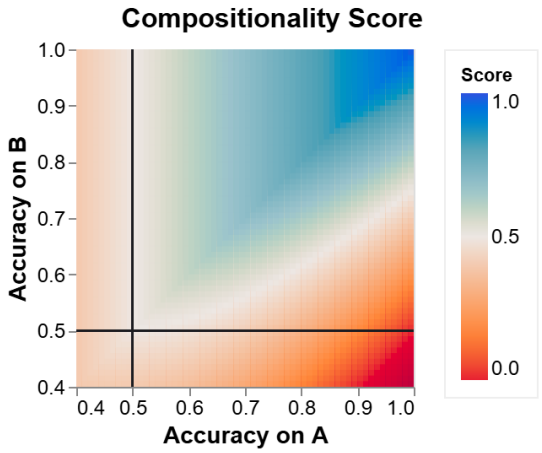
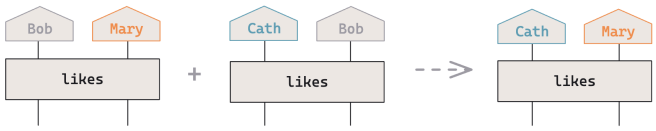


Figure 1.1: $\textsc{cScore}(A, B)$. Baselines at 0.5 are highlighted with black lines.

In the following sections, we explore various choices that can be made when selecting the datasets A and B, in order to target specific aspects of compositionality, formalising the initial adaptation provided in Duneau et al. (2024).

1.2.1 Systematicity



This is one of the better known facets of compositionality thanks to Fodor and Pylyshyn (1988) and their arguments for the compositionality of thought via the compositionality of English. To illustrate with an example, consider the following relatively likely sentence, along with a coarse-grained grammatical type assignment:

$$\underset{\mathsf{NP}}{\textit{Mary}}\ \underset{\mathsf{VP}}{\textit{walked}}\ \underset{\mathsf{P}}{\textit{around}}\ \underset{\mathsf{NP}}{\textit{the block}} \tag{1.1}$$

By replacing each word in (1.1) with a term of matching grammatical type, we can obtain a new sentence that is much less common:

$$\underset{\mathsf{NP}}{\textit{The Martian sous-chef}}\ \underset{\mathsf{VP}}{\textit{duelled}}\ \underset{\mathsf{P}}{\textit{on}}\ \underset{\mathsf{NP}}{\textit{the telephone box}} \tag{1.2}$$

Despite being unlikely to have ever encountered (1.2) before, we can expect any competent English speaker to nevertheless understand the sentence without issue. The reason being (so the argument goes) the compositionality of English: understanding the structure of the sentence and the individual words is already enough to understand the whole sentence - unlike when we encounter a new word, no new learning needs to occur when we encounter a new sentence composed entirely of old words.

A hierarchy of notions of systematicity is introduced in Hadley (1994), depending on how freely terms can be re-combined. In the weak sense, we may only substitute *The Martian sous-chef* for *Mary* if we have previously encountered sentences in which *The Martian sous-chef* occurs as the subject of a verb phrase, whilst in the strong sense it can be substituted anywhere that has the correct grammatical type. To illustrate, we contrast (1.2) with '*The telephone box duelled on the Martian sous-chef*' - while the grammar has been respected, the sentence only makes sense if we personify, and hence change our understanding of, the telephone box.[3]

In this work, a syntax is always strongly systematic; however we can consider the strength of a *model's* systematicity. Applying this distinction to compositional

[3]See Johnson and Journal of Philosophy, Inc. (2004) for a discussion of more problematic versions of this for systematicity in English, and the comments in Section 1.3.

generalisation (reflecting the original usage in Hadley (1994)) allows us to characterise how diverse the samples a model is trained on must be in order for it to be able to successfully generalise. The datasets evaluated in this work did not admit such a distinction; we leave further exploration of this concept to future work.

We will consider cases where e_∇ (the syntax relative to which our task is syntactically compositional) is systematic. Take a set of sentences $A \subseteq \Sigma_L$ and their corresponding diagrams $D_A \in \mathsf{D}_\mathcal{G}(L)$. The systematic closure $\mathsf{sys}(A)$ of A is the set of diagrams in $S_\mathcal{G}$ that can be (recursively) obtained from those in D_A by swapping sub-diagrams with matching type signatures. Thus:

Definition 12 (Systematicity)**.** *A syntax e_∇ is systematic if there exists a proper subset $A \subset \Sigma_L$ where $A \subset \mathsf{sys}(A)$.*

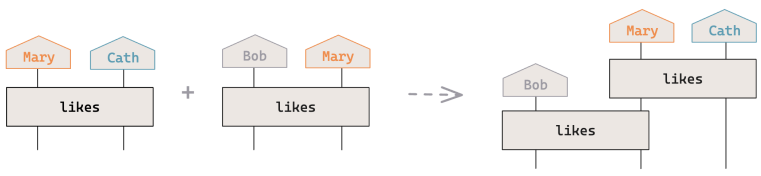
Notice that this definition is slightly different to the usual presentation, which would require a language to be its own systematic closure: $\mathsf{sys}(\Sigma_L) \subseteq \mathsf{D}_\mathcal{G}(L)$. We avoid this definition as it is true by the definition of a syntax: any type-checking permutation that keeps the outer signature of the diagram fixed will remain a valid sentence. We hence will consider a slightly different property, which requires that the systematic closure be non-trivial.

We can then quantify the systematicity of a *model*, relative to a systematic syntax e_∇ and a choice of base set A by considering whether the accuracy drops when moving from the base set to its systematic closure (or, as is more typical to machine learning setups, the elements in the systematic closure that do not occur in the base set).

Definition 13. *A model $\mathfrak{M}$ is ε-systematic over A if:* $\textsc{cFact}(A, \mathsf{sys}(A) \setminus A) \leq \varepsilon$

Of course, this is easy to satisfy if the model performs badly on the base set. The accuracy of the model on the base set remains an aspect that should be maximised too, which is why in practice we maximise the compositional score cScore.

1.2.2 Productivity



Productivity of a language is a method through which to create arbitrarily long sentences using only finitely many symbols. It is best viewed as an existence claim about the rules of L: namely that there is some combination of rules that can be applied recursively to generate sentences of arbitrary length. For example, we can arbitrarily augment a simple sentence by adding extra details to the first noun phrase:

'*The cat sat on the mat.*' (1.3)

'*The great granddaughter of the Marquis of Lewisham's third niece's old tabby cat, who once was the subject of a very long sentence that felt like it would never end (but fortunately will eventually because the both the author and the reader have better things to do), sat on the mat.*' (1.4)

This leads to a simple definition:

Definition 14 (Productivity)**.** *A syntax e_∇ is productive if $|\Sigma_L|$ is not finite.*

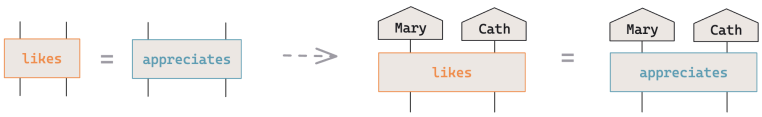
This is impractical, however. Define the productive closure $\mathsf{prod}(A)$ of a certain set $A \subseteq \Sigma_L$ as those sentences that can be obtained by freely applying rules to the atoms found in the sentences of A. This can be obtained by restricting the atoms in the signature $\mathcal{G}$ that generates $S_\mathcal{G}$ to contain only those occurring in the diagrams $e_\nabla|_A \subset \mathsf{D}_\mathcal{G}(L)$. Call this $\mathcal{G}'$. The derived syntax category $S_{\mathcal{G}'}$ will be a sub-category of $S_\mathcal{G}$. The sentences in $\mathsf{prod}(A)$ are then those whose image under e_∇ is contained in $S_{\mathcal{G}'}$.

Note that this may not be a finite set; thus, we limit the 'depth' of the closure by imposing a maximum on the number of rule morphisms that can feature in a derivation - call this $\mathsf{prod}_n(A)$. This in particular distinguishes productivity from systematicity. We can then apply this to a model:

Definition 15. *A model $\mathfrak{M}$ is ε-n-productive over A if:* $\textsc{cFact}(A, \mathsf{prod}_n(A)\backslash A) \leq \varepsilon$

In the case of texts, productivity is easy to achieve by concatenating sentences or other texts together, however this can also occur at the sentence level via the construction of complex phrases. In this work, we will focus on concatenating sentences, in order to keep individual sentences gramatically simple.

1.2.3 Substitutivity



Substitutivity deals with synonyms. The intuition here is that if a model obtains different values for a pair of sentences whose atomic meanings and syntax trees coincide, then the model must be sensitive to something beyond the syntactic structure and atomic meanings, and is therefore not compositional.

The notion seems to presuppose a *syntactically* compositional interpretation in order to determine which atoms are synonyms in the first place, which would make the interpretation substitutive by definition.

In our case, we are interested in *comparing* interpretation functions. We must hence assume the task labels ∇ are not merely semantically, but *syntactically* compositional: $\nabla = e_\nabla; g$, which we can then use to determine the synonyms. We write $\mathsf{syn}(d)$ for the synonyms of a (possibly composite) diagram fragment $d \in \mathsf{Mor}(S_\mathcal{G})$:

$$\mathsf{syn}(d) = \{d' \in \mathsf{Mor}(S_\mathcal{G}) \mid g(d) = g(d')\}$$

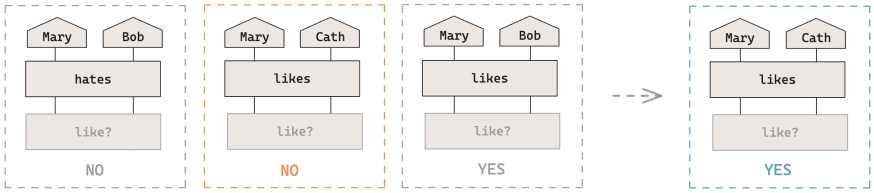
The substitutive closure $\mathsf{sub}(s)$ of a sentence $s \in \Sigma_L$ (and corresponding diagram D_s) is the the set of sentences whose diagrams can be obtained by replacing fragments of D_s with a synonym fragment from $\mathsf{syn}(D_s)$. In order to account for 'nonsense' synonyms (composite terms that aren't ever part of valid sentences but happen to have the same meaning assigned as something else), we further restrict $\mathsf{sub}(s) \subseteq \Sigma_L$.

Definition 16 (Substitutivity)**.** *A model $\mathfrak{M}$ is substitutive if for all $s \in \Sigma_L$, $\mathfrak{M}(s) = \mathfrak{M}(s')$ where $s' \in \mathsf{sub}(S)$*

For the graded notion, we relax the equivalence a little in order to consider accuracies. Fix a base set $A \subset \Sigma_L$ and define $A' = \{a' \in \mathsf{sub}(a) | a \in A\}$

Definition 17. *A model $\mathfrak{M}$ is ε-substitutive over A if* $\textsc{cFact}(A, A' \setminus A) \leq \varepsilon$

1.2.4 Overgeneralisation



This criterion considers a model's ability to memorise exceptions, as opposed to applying a learnt rule to cases where it should not. A compositional model will tend to overgeneralise, whilst a model that is overfitting the data and learning by rote will not.

Unlike the other tests, overgeneralisation is a property that is visible only when the underlying task is almost, but not *quite* compositional with respect to our chosen syntax e_∇. That is, the data involves certain exceptions that do not follow the otherwise compositional pattern. Given compositional task labels ∇, we can derive corrupted labels $\blacktriangledown$, that gives an (incorrect) answer different to ∇ on a random choice of inputs $C \subset \Sigma_L$. $\blacktriangledown$ is therefore *not* semantically or syntactically compositional relative to e_∇ (excepting extremely unlikely situations). We restrict training to a subset A, where $C \subset A \subset \Sigma_L$.

Definition 18 (Overgeneralisation)**.** *A model* $\mathfrak{M}$ *trained against* $\blacktriangledown$ *overgeneralises if* $\mathsf{acc}_C(\mathfrak{M}) \leq \mathsf{acc}_{A \setminus C}(\mathfrak{M})$.

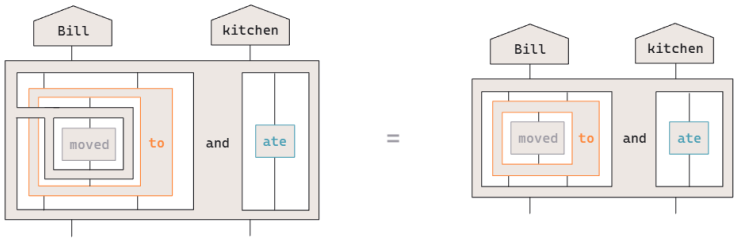
Where acc evaluates the model relative to the true labels ∇ rather than the corrupted ones $\blacktriangledown$. We consider the model to overgeneralise if it performs worse on the corrupted instances than it did on the un-corrupted instances. The graded notion then considers how *much* worse the performance is on the corrupted inputs:

Definition 19. *A model* $\mathfrak{M}$ *ε-overgeneralises over* C *if* $\textsc{cFact}(A \setminus C, C) \leq \varepsilon$

We can also consider a different notion of overgeneralisation, in which the 'exceptions' are introduced in a more structured manner, if we have access to the syntactically compositional decomposition $\nabla = e_\nabla ; g$ of the task labels. We can corrupt only the interpretation of selected diagram fragments in $S_\mathcal{G}$ to obtain a corrupted mapping $g_\blacktriangledown$. This will break the functorality of $g_\blacktriangledown$, and thus $\blacktriangledown = e_\nabla ; g_\blacktriangledown$ will not be syntactically compositional relative to e_∇. Exploring this approach in more detail could provide a route towards handling some non-compositional phenomena while remaining within a mostly compositional framework, and could

be particularly promising for the handling of expressions and idioms where particular word sequences (diagram fragments) might be assigned non-compositional meanings when found together but behave compositionally otherwise. While a new atom could always be introduced for such exceptions in order to recover compositionality, this artificially bloats the syntax; doing without could allow for a simpler presentation. We leave this for future work.

1.2.5 Localism



Localism is a property that can be expanded in two ways. The first, much like substitutivity, is tied to having some base interpretation that tells us which terms we might take as equivalent, while the second can be seen as a constraint on the kind of morphisms we allow in $\mathcal{G}$. To illustrate the first via an example, a fragment of language often taken as non-local is that of epistemological claims. Consider the following two sentences:

$$\text{`}\textit{Mary believes that grass is green.}\text{'} \tag{1.5}$$

$$\text{`}\textit{Mary believes that grass reflects light in the 550 nanometer range.}\text{'} \tag{1.6}$$

Although *is green* and *reflects light in the 550 nanometer range* are synonyms according to their English definitions (in that they identify the same collection of things), it is entirely plausible that Mary doesn't know which wavelengths of light look green. Thus (1.5) might be true while (1.6) is false, suggesting that the *whole* sentences should not be synonyms if their meanings are to capture this difference.

However, an interpretation that composes meanings *locally* is not able to distinguish (1.5) and (1.6): both sentences share the 'outer' structure 'N *believes that* S', and the meaning of N (*Mary*) and S (*grass is green*) are equal. Put another way, when assigning meaning to a sentence of the form '*Mary believes that* S', we need to consider what S means *to Mary* rather than what it means *in English*,

which might require some way to 'reach in' to the sub-structure of S to make the appropriate adjustments.

Formalising this intuition, we consider an interpretation or model to be local if the meaning it assigns to sub-expressions does not depend on the overall expression they are part of. Once again, suppose we have some syntactically compositional labels, such that $e_\nabla : \Sigma_\mathcal{L} \to S_\mathcal{G}$ is our preferred syntax for determining sub-expressions. Localism then this becomes a very strong claim:

Definition 20. *A model $\mathfrak{M} : \Sigma_\mathcal{L} \to \mathcal{M}_L$ is local if it is syntactically compositional relative to e_∇.*

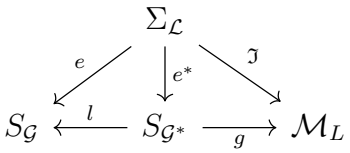
Indeed, since localism requires a notion of the interpretation assigned to sub-expressions (or sub-diagrams), $\mathfrak{M}$ must be analysed via $S_\mathcal{G}$ for its action on diagram fragments to be well-defined. The functorality of the interpretation then takes care of the rest of the requirements. Weakening this notion slightly, we may also consider *semantic* locality, in which the model need only be semantically compositional. This notion is more in line with that considered by Hupkes et al. (2020), since it allows for a broader range of model architectures.

In this work however, we will focus on functorial models, and so introduce a notion of locality at the syntax level as well. Intuitively, we want to capture the idea that a syntax with a non-local interpretation can be upgraded to a 'non-local' syntax with a local interpretation by allowing either the atoms or the rules to be duplicated and indexed to reflect the surrounding grammatical structures. Thus, we might distinguish *light in the 550 nanometer range*, *light in the 550 nanometer range*$_{Mary}$ and *green*$_{Mary}$ if duplicating atoms. Alternatively, duplicating rules might lead to a series of production rules for 'N *believes that* S_i', where each S_i has a fixed inner structure.

Definition 21 (Localism)**.** *A syntax $e^* : \Sigma_\mathcal{L} \to S_{\mathcal{G}^*}$ is non-local relative to $e : \Sigma_\mathcal{L} \to S_\mathcal{G}$ if we can partition the morphisms and types of $\mathcal{G}^*$ into equivalence classes, indexed by the items in $\mathcal{G}$. We write $l : S_{\mathcal{G}^*} \to S_\mathcal{G}$ for the functor that sends each base morphism in $\mathcal{G}^*$ to the representative of its equivalence class in $\mathcal{G}$.*

Suppose we have an initial interpretation $\mathfrak{I}$ that is non-local relative to e (and thus cannot factor through $S_\mathcal{G}$). Then we can construct a non-local syntax e^*

relative to which $\mathfrak{M}$ *is* local:



We can also cheat localism somewhat via distributivity in the semantics: rather than duplicating syntax entries, we can add redundancies to the meaning space directly, and allow the outer context (if present) to disambiguate which is required when composing them.

Informally, a graded notion of locality should quantify the redundancy in $S_{\mathcal{G}^*}$. We shall be less concerned with this measure in this work, as, unlike the others it is more directly under our control due to its dependence on the syntax. Indeed, the models that we will be investigating are necessarily local, so any amount of non-localism must be added in at the syntax level explicitly. We leave the investigation of non-localism arising in the meaning category to the analysis of final models rather than measuring this specifically.

1.3 Required assumptions

The task setting at hand - reading comprehension tasks, expressed in English - requires certain assumptions if it is to be solved in a useful way by a compositional model. To break this down for the general case, we would like both English, and the task of question answering to be compositional relative to some nice rules. In the language used in this paper, this means finding a pair of syntax embeddings e^* and e_∇ that can witness the compositionality of English meanings ($\mathfrak{I}_L^*$), and question answering tasks (∇), respectively. This is however, unlikely to be the case.

1.3.1 English is compositional

English is, most likely, not quite compositional. There are many well debated phenomena that would suggest otherwise, such as indexicals (*here*, *I*), idioms (*kicked the bucket*), or noun modification (identifying *the dog spoon* amongst a set of spoons each used by a different animal, as opposed to amongst a set of spoons with animal engravings). See e.g. Partee (1984a) for a summary, and Johnson and Journal of Philosophy, Inc. (2004) for a particular focus on systematicity.

The common theme among many of these issues appears to be a notion of contextuality - the meaning of the words noted above can change depending on the circumstances in which they are uttered. But, this doesn't *have* to be a barrier to a compositional account. Recanati (2012) argues that the atoms of meaning should be upgraded to functions that can modify base meanings with a given context. In a similar spirit, the DisCo- family of models we will study in this thesis has a notion of context dependence built in, by allowing the meaning of individual words to be ambiguous, such that the composition with the rest of the context is what disambiguates each word simultaneously (Grefenstette and Sadrzadeh, 2011; Kartsaklis and Sadrzadeh, 2013).

Another source of concern is pragmatics: when what is said isn't what is meant. Damning with faint praise and irony are examples of this, where the intended meaning is supplied by what is *not* said, or is actually the opposite of what is said (Grice, 1975). In such cases, an ambiguity based account seems much less likely to succeed, or be practical. Analyses of what is meant in such situations also tends to require some shared background knowledge, that does not need to be explicitly stated, but nevertheless contributes to the context.

For our current purposes then, we will assume that we can restrict ourselves to a sub-language of English that *is* compositional in an interesting way, effectively by simplifying and removing all problematic elements. In particular, we will be controlling the specific ways in which the fragment of language we use is compositional, in order to test these aspects directly, as described in Section 1.2 and Chapter 3.

This does not have to be as damning an assumption as it might initially seem. For one, a lot of English is pretty close to being compositional. Though there are likewise many areas that tend to be less so (poetry, reported speech, ...), things like reading legal documents, logical puzzles, and formal languages likely admit non-vacuous compositional interpretations. Moreover, the benefits of compositional generalisation and understandability are nothing to frown upon in a setting where training or data collection is expensive - in such cases, perhaps an approximate, but compositional solution would be preferred.

Secondly, we have been intentionally very inclusive about what sorts of rules are permitted as part of a syntax; compositionality is only well defined with respect to a syntax, and as pointed out before we can certainly come up with rules that make being compositional relative to this syntax a vacuous property. When we say that English is not quite compositional, what we actually mean is that the grammar with respect to which English might be made compositional isn't one

that satisfies all of our implicit desirability criteria. Though we restrict ourselves to a safe corner for the purpose of this proof-of-concept work, the approach itself has no equivalent constraint.

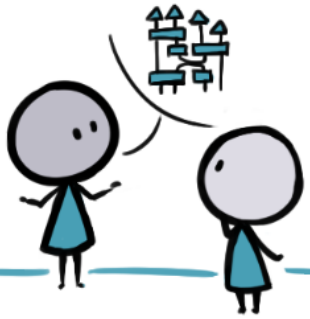
1.3.2 Question answering is compositional

Again, this is unlikely to be true in general, for example when asking questions with contextual answers ('*Is the Queen of Hearts correct?*' - about what?), or indeed when the answers are open-ended ('*How might one use compositionality to solve reading comprehension tasks?*') so we will restrict our attention to a class of compositional tasks.

The tasks we consider are designed to require no background knowledge to solve, so the contextuality issue becomes irrelevant. The crux of the issue is then whether we expect that composing the question with the rest of the context is all that is required to derive the answer from the meaning of the context and question individually. Since the question is a separate sentence, the only compositional rules that apply are the rules governing the grammar of the question itself, and the rules governing inter-sentence composition. These include anaphora and ellipsis (which do not belong in the fragment we are considering here) and co-reference resolution (which we do care about). For the tasks considered, it does seem to be the case that the only relevant structure that allows us to derive the final answer from the meaning of the context and the meaning of the question, is specifically the co-reference information which tells us which part of the context the question applies to.

An important caveat to consider here is the space of possible answers, which are often contextual. When attempting to answer '*What colour is the ball?*' one must know what the possible colours to chose from are - in the context of pool this might be {*white*, *yellow*, *red*, *black*} whilst for an artist trying to paint a still life the selection might be much more detailed.

The issue is less prominent when it comes to binary questions like '*Is Mary in the kitchen?*' since they seem at first glance to admit exactly two possible answers: *yes* or *no*, however we can still imagine circumstances where we would rather answer *maybe*, or *definitely not*. This problem is thankfully mitigated in the context of training a model for a specific task, as the space of possible answers is generally given as part of the task definition. We hence chose to hard-code these decisions as part of the task implementation.



CHAPTER 2

DisCoCirc

In this work, we will take the DisCoCirc framework as a starting point for building models, as this gives us *syntactic* compositionality by construction. Solving a compositional task then becomes a question of whether the model can be trained to be *semantically* compositional too. This chapter first introduces the background structures required to express DisCo models, then presents both DisCoCirc (Coecke, 2020) and its predecessor DisCoCat (Coecke et al., 2010) in a unified way. Finally, we consider how the basic formalism can be adapted for the task of question answering considered here.

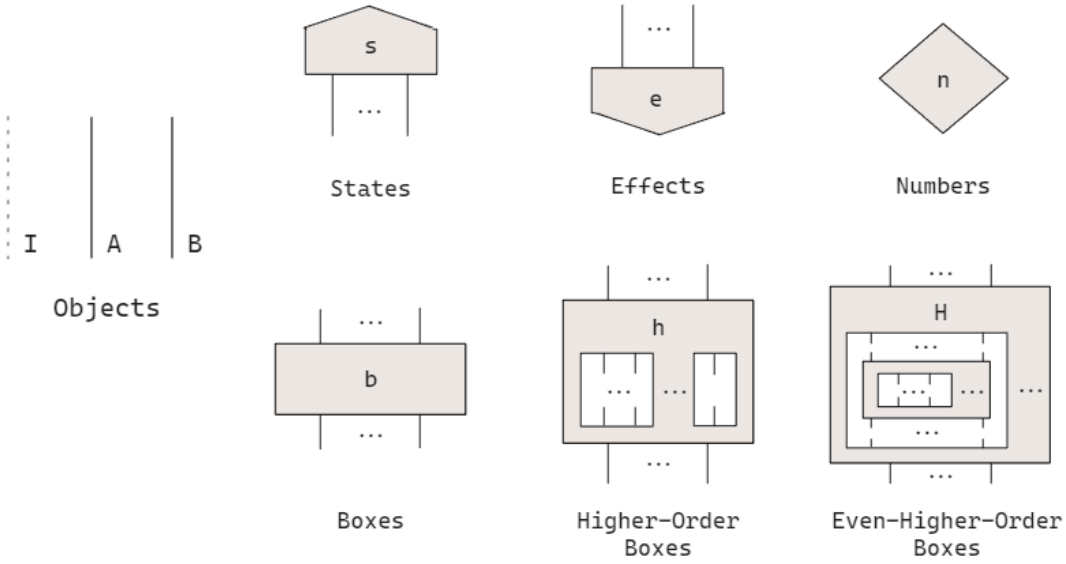
2.1 Notation

The models presented in this work admit natural graphical representations by virtue of living in monoidal categories. Here, we give a brief overview of the notation and structures used in this work, but refer the interested reader to Piedeleu and Zanasi (2023) for a more comprehensive treatment of the relevant concepts. All diagrams presented in this chapter should be read top to bottom. (In Chapter 4 some diagrams will be read left to right.)

2.1.1 Monoidal category

We assume a monoidal category for the diagrams. This means we can place objects next to each other as well as composing them sequentially. The generators are illustrated in Figure 2.1: Objects are denoted by wires. The monoidal unit (`I`) is drawn as a blank wire, or empty diagram (shown dashed as a visual indication).

Morphisms are drawn as boxes over the input/output objects. The identity morphism $I_{\mathtt{A}}$ is drawn as a wire. We distinguish morphisms to and from the monoidal unit as states and effects respectively. Morphisms $n : \mathtt{I} \to \mathtt{I}$ are called numbers. A diagram with no open wires is also a number.



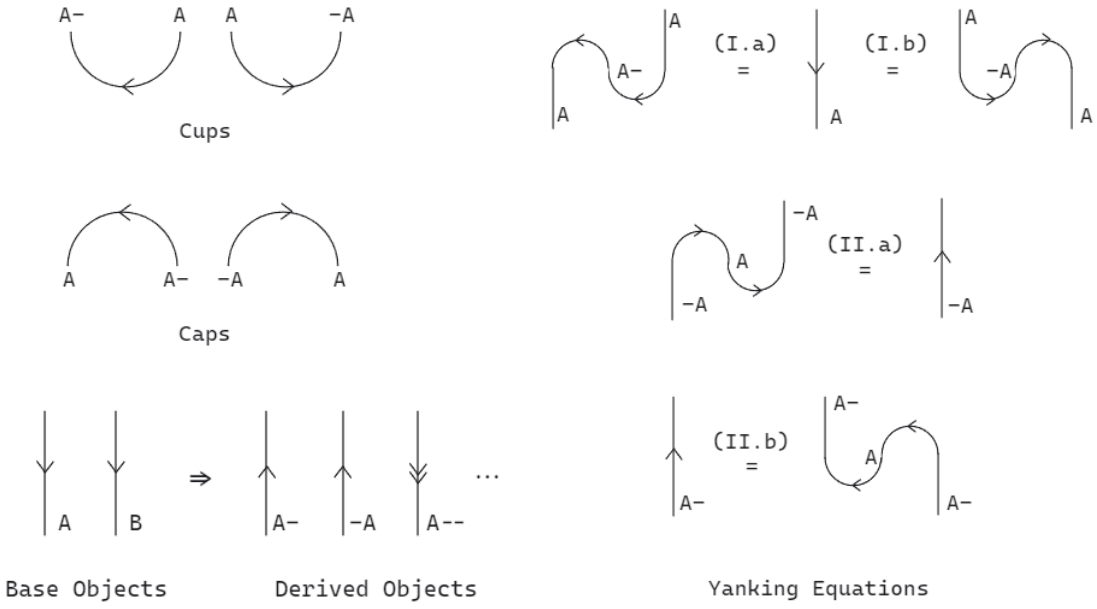
(a) Objects (b) Boxes

Figure 2.1: Base generators of a monoidal category.

Finally, we also introduce the notion of higher-order morphisms - frames - which take boxes as input rather than objects. Similarly, 'even-higher-order' morphisms act on higher-order morphisms. We draw frames as boxes with holes, where the shape of the hole dictates the type of generator expected. In this work, we will consider only boxes with a single hole, and symmetric input and outputs. We write $HO(n, m)$ for a higher order box with n outer dangling wires on each side, and n inner dangling wires on each side. A frame with filled holes is just a normal box. In this work, we will remain largely agnostic about how frames should be formalised, as they can (with some restrictions) be re-written in terms of only boxes (as explored in Section 2.1.5), which is more convenient in practice. Some available options are supermaps in the sense of Hefford and Wilson (2024), with the additional restriction that wires may not cross box boundaries (Tull et al., 2024), or associative n-categories (Reutter and Vicary, 2019) as presented in Wang-Mascianica (2023).

2.1.2 Rigid

A rigid category is monoidal, with the addition of some extra structural morphisms known as cups and caps, as illustrated in Figure 2.2a. Each wire is given a *direction* (drawn as an arrow), as well as a *handedness* (indicated by the side on which the minus (−) sign occurs). This allows us to derive new objects from the base generators, that relate to the base object via the special states and effects called cups and caps. These satisfy the 'yanking equations' shown in Figure 2.2b.



(a) Generators

(b) Axioms

Figure 2.2: Extra generators and rules introduced by rigid categories.

Compact closed

When the left and right adjoints are equal ($A^* := A^- = {}^-A$), the category is known as compact closed. This allows us to simplify the presentation by omitting the arrows on the wires.

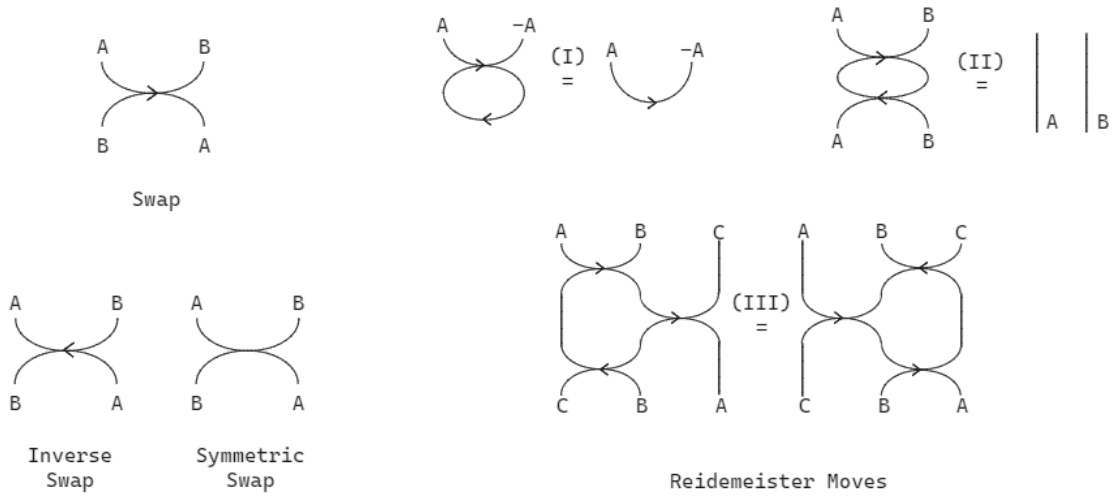
2.1.3 Braided

This structure allows us to cross wires over one another, via the addition of a new swap box. We direct the swap to distinguish it from its inverse, as pictured in Figure 2.3. The Reidemeister moves (Figure 2.3b) determine how the swap interacts with other structural morphisms: if a cap is present, the braiding interacts

with it as per (I). (II) establishes the inverse while (III) allows one swap to pass 'underneath' another.

Symmetric

A symmetric category has a self-inverse braiding, which we draw undirected.

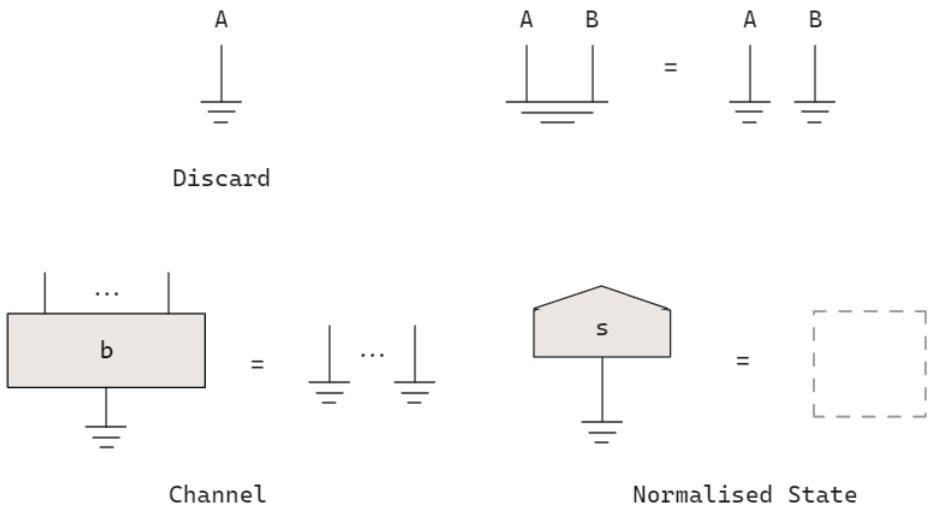


(a) Generators (b) Axioms

Figure 2.3: Extra generators and rules introduced by braided categories.

2.1.4 Discard

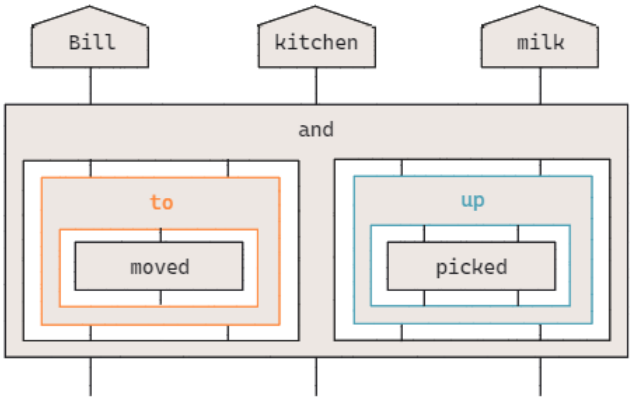
A discard category is a symmetric monoidal category that additionally has a family of special discard morphisms for each object. The effect is drawn as three horizontal lines, and is natural with respect to the monoidal product.



A state is called normalised when discarding it leaves us with the identity. Morphisms that preserve the discard are known as channels.

2.1.5 Frame expansion

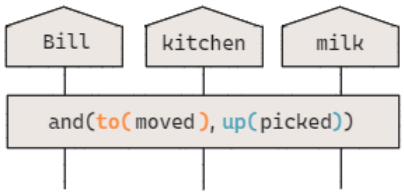
Formalising frames tends to require elaborate machinery that is not always practical from an implementation perspective. In certain cases, however, they can be re-written into frame-less diagrams. We illustrate the various strategies via an example:



This is the DisCoCirc parsing for the sentence '*Bill moved to the kitchen and picked up the milk.*': the outer frame has two holes, with nested frames inside each. The inner diagrams do not always have the same number of wires as the outer diagram. The boxes are coloured as a visual aid.

Pancake

The most basic expansion simply expands the box name in terms of its insides. This requires no further structure, however loses all of the compositional benefits of the model within the frame.



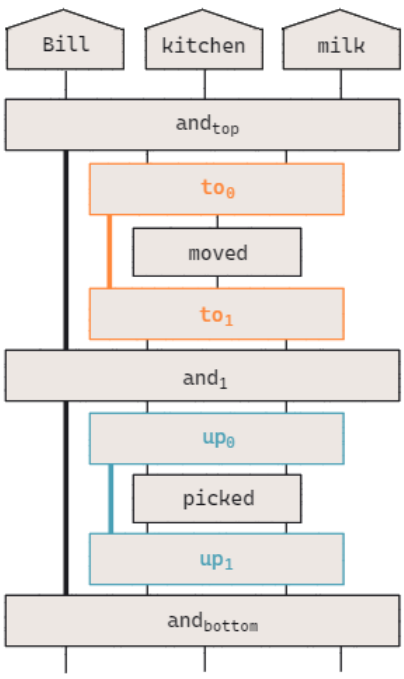
Pancake

Kebab

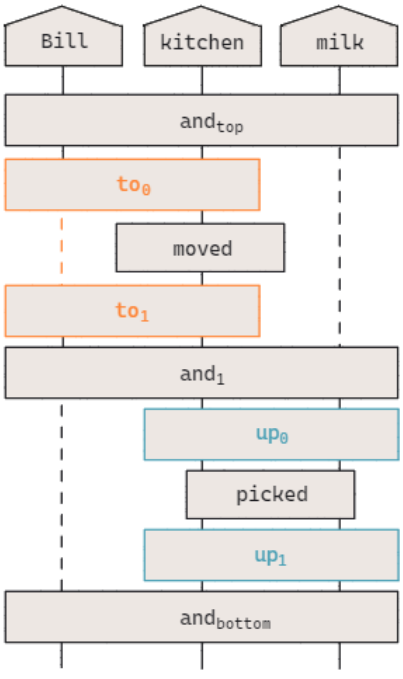
A somewhat more sophisticated notion might instead break the outer frame into linked pieces. In the example shown, each higher order box introduces a new thick wire (coloured to match the box it links) of a new type $*$. Boxes with multiple holes insert an intermediate box between the expansion of each hole. The holes are inserted in order from left to right. Note that rather than stacking the holes vertically, we might also have stacked them horizontally, and adjusted the wirings for the *and* boxes accordingly.

Sandwich

Similar to the kebab expansion, we introduce the sandwich. In this case, the number of wires present inside the holes is made to match the wires present outside, and thus will stay constant throughout the diagram. Extra wires are shown dashed. This particular expansion requires some extra clauses to handle frames that have more wires on the inside than outside - in such cases the selected convention is to suppress the extra wires as described in Laakkonen et al. (2024). This wire suppression can lead to some boxes having a different type depending on their location within the diagram, which in turn makes the resulting model non-local.



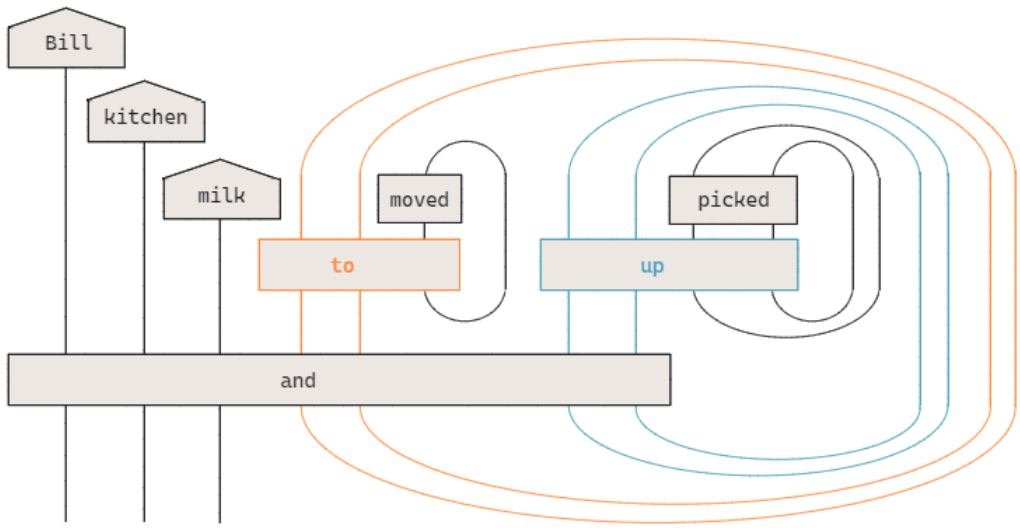
Kebab



Sandwich

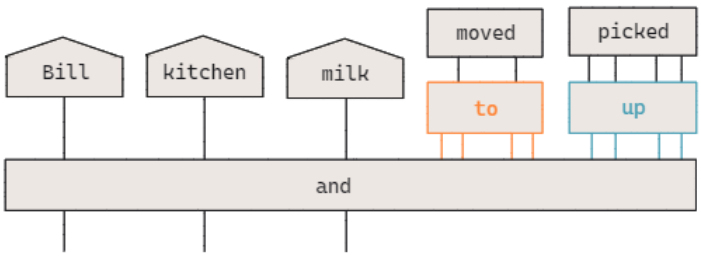
Onion rings

The onion-ring expansion remains most faithful to the idea of a higher order map: each frame is re-written as a flat box over the same wires. Each outgoing wire that was originally inside a hole is bent upwards using a cup, whilst the inputs to the inner diagram are bent downwards via a cap. The wires are coloured as a visual aid.



Onion Rings

We can then absorb the cups and caps into the boxes by way of the map-state duality, shuffle the wire orders a little, and rewrite the diagram in a friendlier format:



Tiered Cake

Tiered cake

Given that the final format has no overt cups and caps, we could also expand the frame to this directly, however this incurs the drawback that boxes occurring inside frames will have a different type signature to those outside, once again

sacrificing locality.

2.2 DisCo- formalisms

Following the presentation of a language in Chapter 1, the DisCo- formalism recipe can be split into three parts: syntax - defining $\mathcal{G}$ and $S_{\mathcal{G}}$; rewrites - which help define the labels ∇ we target for semantic compositionality; and semantics - defining the meaning category $\mathcal{M}$. These parts are then tied together by a model $\mathfrak{M}$.

Syntax

As per Definition 3, the syntax forms a monoidal category (with extra structure depending on the exact formalism). The objects are given by a set of chosen grammatical types, whilst elements of the vocabulary are represented as boxes acting upon these. The grammatical rules dictating how to build sentence diagrams are also included here, as well as the basic structural rewrites. Crucially, for DisCo- formalisms, the grammatical rules that guide the construction of the diagrams are all structural; this means they are typically not drawn explicitly as boxes, and instead merely control how the atoms connect to one another.

Rewrites

The rewrites of the DisCo- formalism dictate what form the task labels should take. Going beyond the basic grammatical rules for generating sentences, we will often add extra rewrite rules not captured by the syntax, intended to express equivalences between certain expressions. Some of these may be purely syntactic, such as the equivalence between active and passive forms of a sentence - '*The farmer's sheep gave birth to a lamb.*' and '*A lamb was birthed by the sheep that belongs to the farmer.*'. Meanwhile others are constraints on the interpretation (often taken in a specific setting) such as asserting spatial axioms like '*turns around then turns around*' is the same as doing nothing. This can also include synonym replacements.

For known rewrites, these can be explicitly encoded as part of the semantic functor or model we build, however, in general, the rewrites are not fully specified, as this would require axiomatizing meaning. In certain specific cases or for very narrow tasks, one can find a list of axioms that appropriately constrain the meaning function, however for practical purposes, we expect to learn or infer

many of these rules implicitly. Paraphrase information can be viewed as a natural (though highly redundant) way to encode this information, so is an ideal candidate as the rewrite set for a generic model. Similarly, when training a model for a specific task, the training data itself can be viewed as implicitly encoding rewrites via the task labels ∇.

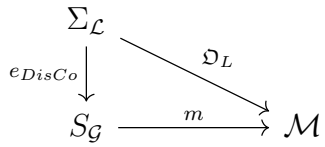
Semantics

The semantics of the model lie in a different category. We explicitly enforce a functorial transition from syntax to semantics, such that the induced models are syntactically compositional, in that they necessarily factor through the chosen DisCo- grammar. [1]

The choice of category is left open - previous works have used *Rel* (de Felice et al., 2020; Duneau, 2021b), the category of completely positive maps over finite dimensional Hilbert spaces $CPM(FdHilb)$ (Coecke and Kissinger, 2017; Bankova et al., 2016; Cuevas et al., 2020; Lorenz et al., 2021), as well as semantic spaces following Gärdenfors (2011). $CPM(FdHilb)$ with tensor product $\otimes$ is considered particularly appealing due to both its compact closure and for allowing vector-based distributional semantics, which are well suited to machine learning. We will also consider $\mathcal{NN}$, a category of neural-networks (defined in Section 4.2).

Models

A DisCo- model $\mathfrak{M} = (e_{DisCo}; m,\ \mathfrak{M}_{\mathsf{imp}})$ is a functor from the DisCo- category of diagrams $S_{\mathcal{G}}$ to the semantic category $\mathcal{M}$. An ideal model additionally makes the following diagram commute, for a particular task oracle $\mathfrak{O}_L$:



That is, the model matches the task oracle $\mathfrak{O}_L$ (that respects all the implicit rewrites) over all inputs in Σ_L. Training thus attempts to minimise the differences between $\mathfrak{M}$ and $\mathfrak{O}_L$.

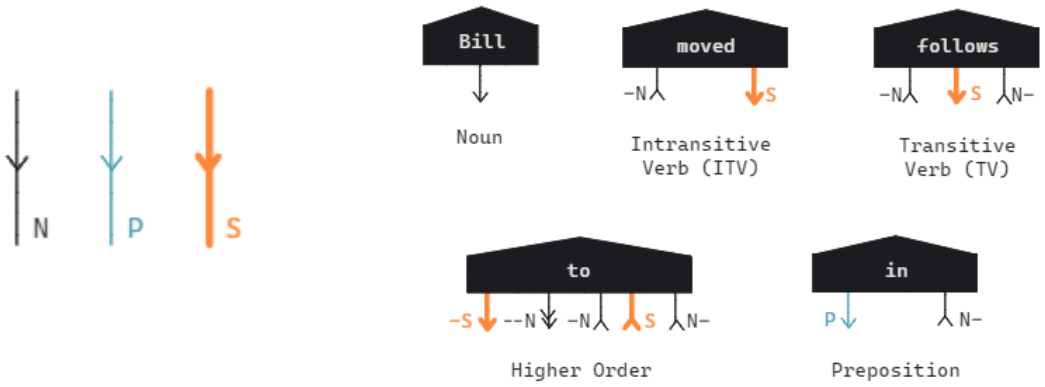
[1] One might consider other architectures like neural networks (which can also be expressed categorically) to be necessarily syntactic too, in some sense. The crucial difference is that the structure relative to which the DisCo- models are syntactically compositional is one along which we can also decompose the task oracle.

2.2.1 Frameworks

DisCoCat

The DisCoCat formalism (Coecke et al., 2010) was introduced as a way to construct sentence embeddings that can be directly compared with each other, stemming from the insight that pregroup grammars (Lambek, 1999, 2006) can be elegantly expressed using a rigid category, and that distributional meanings naturally live in a vector space, which shares the same categorical structure.

Syntax The generators for DisCoCat are displayed in Figure 2.4. Not shown are the structural morphisms for a rigid category (cups and caps), as well as an optional braiding. A special feature of DisCoCat is that the grammatical rewrites in $\mathcal{G}$ are exactly represented by a series of nested cups and identities, such that the final diagrams in $S_{\mathcal{G}}$ are states with a single output type `S`, that decompose as a series of atom states (for each of the constituent words), linked together by structural morphisms as rules. More recently, Yeung and Kartsaklis (2021) have used a CCG (Combinatory Categorial Grammar) parser to obtain DisCoCat diagrams, which additionally requires a braiding.



(a) Objects (b) Example boxes

Figure 2.4: DisCoCat syntax. DisCoCat boxes will always be filled in block colour to distinguish them from the outlined DisCoCirc boxes. (a) For this work, we distinguish three grammatical objects: nouns (`N`), prepositions (`P`) and sentences (`S`). (b) The boxes assume underlying pregroup types over the base objects, and are all presented as states.

Sometimes, it is inconvenient for the diagrams to have quite so many cups: we can define a functor circify over DisCoCat diagrams that removes them. This will in general introduce frames, however allows DisCoCat diagrams to be embedded

in a non-rigid category, which can be of significant practical interest. The intuition for this is as follows: each adjoint wire can be 'bent upwards' by adding an extra cup and cap. We then redefine the box to absorb the cap, and yank out the initial grammatical cup with the paired cap just introduced. We will in general require a braiding for this step, however once complete all wires should flow down the page without caps or cups. This process can be viewed as somewhat adjoint to the tiered cake frame expansion from Section 2.1.5. An example parsing is shown in Figure 2.6.

Rewrites Much work has been done to simplify the representation of certain words that are considered to contain only syntactic content. For example, with the addition of spiders (special dagger commutative Frobenius algebras), we can expand relative pronouns (Sadrzadeh et al., 2013, 2016) and conjunctions (Kartsaklis, 2016).

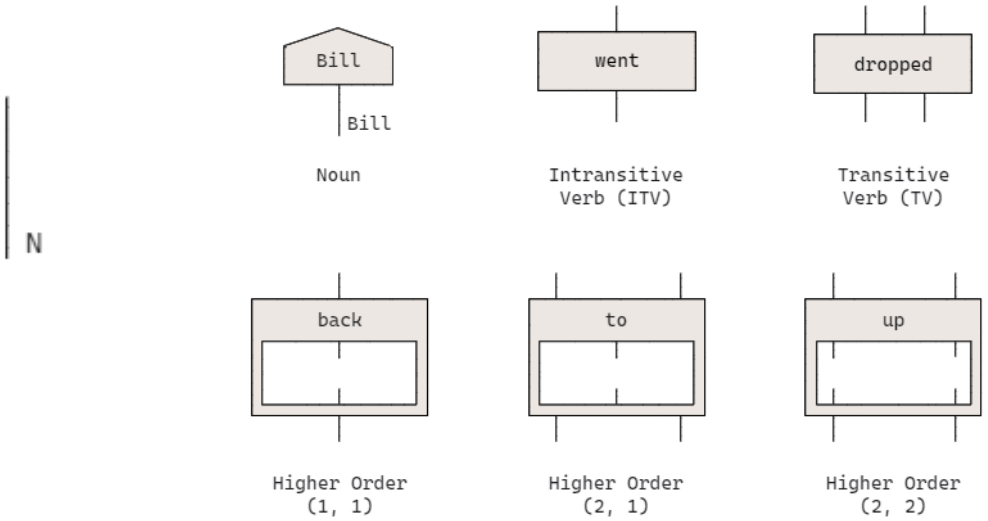
Semantics DisCoCat was originally conceived as a way to allow the meaning of entire sentences to be compared to one another on equal footing. In this sense, the grammar acts as a way to combine the meaning of individual words into that of an entire sentence.

DisCoCirc

The DisCoCirc framework builds upon DisCoCat, with the intention of additionally allowing entire sentences to be composed. In general, each sentence will give rise to a DisCoCirc diagram. The diagrams for multiple sentences can then be composed by making use of inter-sentence grammatical rules, such as co-reference information as well as anaphora and ellipsis resolutions. The initial presentation in Coecke (2020) upgrades DisCoCat diagrams to DisCoCirc, whilst Coecke and Wang (2021) and Liu et al. (2023) construct the diagrams directly via an intermediate grammar.

Syntax We represent the syntactic structure as a symmetric monoidal category with a single (non-trivial) object $\mathtt{N}$ (the noun type), and boxes given by the various words in our vocabulary. The shape of each box depends on its grammatical type - a transitive verb is a box $tv : \mathtt{N} \otimes \mathtt{N} \to \mathtt{N} \otimes \mathtt{N}$ with two wires going in and two wires going out, whilst adverbs are higher-order boxes $adv : V \to V$ for V the set of verb boxes, which we draw as frames. Nouns take a special place in DisCoCirc: each instance of $\mathtt{N}$ is labeled by a specific noun (eg *Bill*), and they also have an associated state $s_n : \mathtt{I} \to \mathtt{N}$ to introduce the labeled wire.

The base grammatical category for DisCoCirc must be braided (usually we assume symmetrically so for convenience), as we must be able to rearrange the order of the nouns as part of connecting sentences together. Figure 2.5 presents the generators for DisCoCirc. For a more complex fragment of language, we expect to encounter more higher order boxes, including boxes with multiple holes, however we show only the shapes required for the tasks considered in this work and refer the interested reader to Liu et al. (2023) for further examples.



(a) Objects (b) Example boxes

Figure 2.5: Syntax for DisCoCirc. (a) There is only a single object type: `N`, and the monoidal unit `I`. (b) The boxes are distinguished first by shape, then by name. Higher order boxes are drawn as frames.

Semantics Rather than viewing DisCoCirc diagrams as static encodings as was the case with DisCoCat, the perspective shifts to viewing them as processes that happen to nouns, or a collection of 'main characters'. We hence lose the consistency of a unique sentence type, but gain the ability to compose sentences together in order to handle paragraphs. This is a modelling choice that echoes the choices typically made in other domains such as physics, where theories are presented as a pair of ontology and dynamics: the ontology specifies what sorts of things exist in the world, while the dynamics specify what these entities can do and how they may interact with each other. It is typically assumed that the postulated entities have properties, whose evolution is guided by the dynamics.

There are however other ways we might look at things, by shifting what we consider primitive. Relational Quantum Mechanics (Rovelli, 2021) is an interpretation

of quantum mechanics that takes events - or *facts* about an interaction between some systems - as the primitive entities. The dynamics are then concerned with the probability of a certain fact occurring, given a previous fact.

Analogously we might have picked events as the objects flowing through the wires of a DisCoCirc diagram, and described everything else as a process acting upon and relating these events to one another. This choice impacts what sorts of tasks are easy for the model to solve and which ones become more complex. In particular, taking the usual noun-centric view, querying a text about properties of a particular noun or the relationship between nouns is straightforward, while it may be more difficult to extract information about events: '*Does Jill have a twisted ankle?*' versus '*How many times was a house exited?*'.

Having selected nouns as the basic entities tracked by our model, it pays to be a little more specific about how exactly these nouns should be selected. In particular, typical presentations of grammar tend to have more than one noun-like type. Indeed, even in quite simple situations, we might come across multiple types of noun: pronouns (*I*, *you*, *they*), proper nouns (*Mary*, *Tesco*), common nouns (*ball*, *house*), and even composite nouns (*St Paul's Cathedral*, *The king of France*).

From the perspective that a DisCoCirc diagram is a model of the text's world, an intuitive choice for what should count as a noun is a discourse entity of Discourse Representation Theory (Karttunen, 1969; van Eijck, 2006). This provides an account of how the entities present in a text should be identified across sentences, as well as identifying the scope during which they are relevant. Indeed, some entities may only feature within a context once, perhaps to provide further information about a different, more persistent entity. Such nouns then might enjoy a temporary wire that is discarded once they fall out of scope. A later (unrelated) use of the same word would then re-introduce the noun with its own state rather than connecting to the previous wire like a discourse entity.

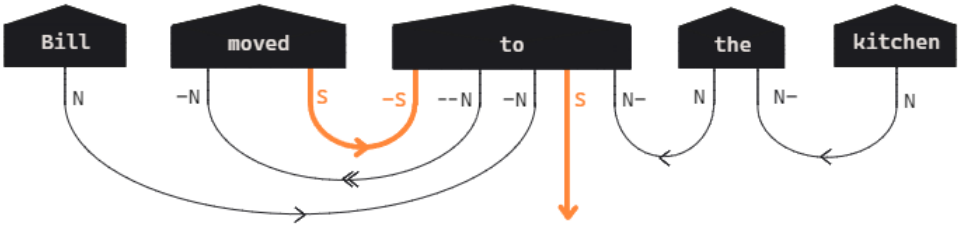
When applying the DisCoCirc formalism, we must take care to distinguish between the order in which events in a text occur (at least relative to each discourse entity), and the order in which they are presented. For the purposes of this work, we shall assume these equivalent. Interestingly, Partee (1984b) draws parallels between tense and pronoun resolutions, giving an account of discourse *events* that mirrors that of discourse entities. This might then suggest using an event-based perspective within DisCoCirc, or perhaps combining both in a unified space.

Rewrites In many cases the rewrites used in DisCoCat can be ported over to DisCoCirc. We will also make use of certain task-specific rewrites described in Chapter 3.

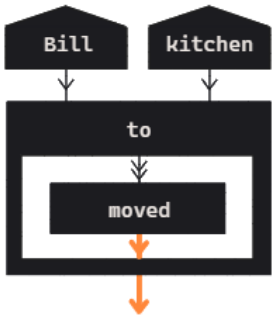
Examples

Figure 2.6 and Figure 2.7 show example sentences parsed as DisCoCat and DisCoCirc diagrams respectively. Notice that DisCoCat diagrams cannot be composed other than placing them next to each other (with final type $\mathtt{S}^n$) whilst DisCoCirc diagrams plug together.

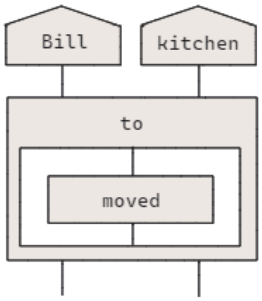
'*Bill moved to the kitchen.*'



(a) DisCoCat parse



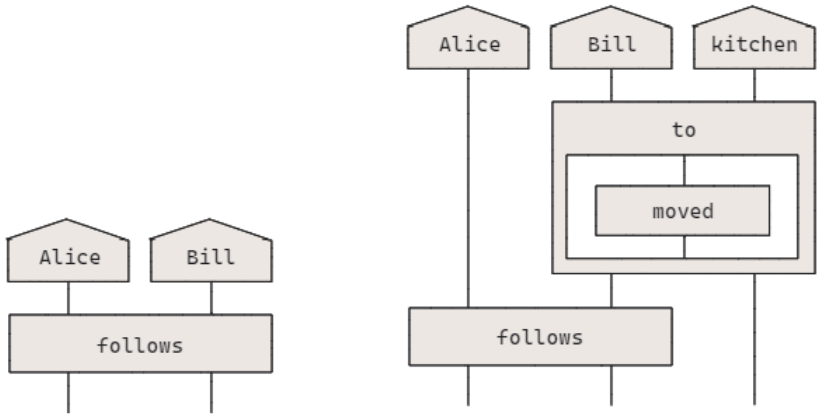
(b) circify DisCoCat



(c) DisCoCirc

Figure 2.6: Parsing a sentence in the DisCo- frameworks. (a) The standard initial DisCoCat parsing: on the top layer the box assignments for the sentence are arranged in the order in which they appear. Below, the wires are connected with cups. Valid sentences can be connected (without crossing wires) such that there is a single dangling $\mathtt{S}$ wire remaining. (b) In many cases, it is convenient to rewrite the initial parsed diagram into its circify-ed format. We will generally also apply syntactic rewrites during this step to simplify the diagram, such as replacing *the* with an identity. (c) The same sentence parsed with DisCoCirc (with rewrite rules applied). Notice that this form is very similar to the circify-ed DisCoCat form, where the sentence wires have been replaced with a series of noun wires.

'*Bill moved to the kitchen. Alice follows Bill.*'



(a) (b)

Figure 2.7: Composing sentences in DisCoCirc. (a) The isolated parsing of the second sentence '*Alice follows Bill.*'. (b) The second sentence is appended to the diagram of the first sentence by connecting wires corresponding to nouns that re-occur (*Bill*), and dragging the state initialisations to the top of the diagram.

Compositionality

DisCo- models are *syntactically* compositional (with respect to a fragment of English) by definition - what needs to be learnt is compatibility with the rewrite rules and *semantic* compositionality with respect to the task oracle g. This freedom between the syntactic and semantic compositionality of the model allows for a lot of versatility. Given a sufficiently rich meaning category, a syntactically compositional model can approximate a non-compositional interpretation as closely as desired. For example, extending the way lexical ambiguity is handled in Piedeleu et al. (2015), atoms can be given a series of meanings that are only disambiguated fully by their final context. In this way, the model learns to encode a lookup-table of answers. The main challenge when training a model to match a compositional g is then to ensure that the model is making correct use of its internal syntactic structure, rather than simulating a non-compositional function.

2.3 Question answering

While the DisCoCirc formalism described so far lends itself well to storing information, and DisCoCat allows computing a fixed function of the meaning of a single sentence, we have not yet seen how extract information from an entire story in a more flexible way. Laakkonen et al. (2024) provide a selection of NLP tasks

DisCoCirc is naturally suited to, including question answering, which we focus on here. We present two main ways of doing this, both rooted in the same idea that questions are also texts, with their own encoding, and thus can potentially be appended to a text themselves, or compared to other texts via a generalised notion of inner product.

2.3.1 Scope

Handling questions in this way only allows for questions where the possible answers have been enumerated in advance, as we do not yet have an account for *generating* novel sentences. Such questions can all be framed as a classification task, where the model must select the correct answer amongst a list of choices.

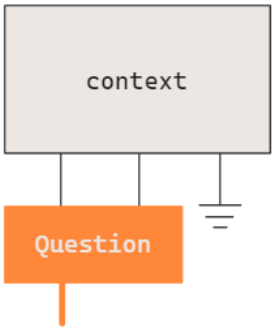
Previous work (Duneau, 2021b) considered problems with a fixed question across all instances of a task, such that the expected answer could be derived directly as the embedding of the context into the meaning category. Indeed, the expected answer was simply a complete description of the final state of the characters in the text, which is exactly what a DisCoCirc model provides. In contrast, there are many possible questions we might ask of a given context for the examples considered here. Though a complete description of the final state of the characters may be sufficient to answer the question, it is generally neither minimal nor efficient to compute. As such, in this section we will explore some methods for extracting information from the DisCoCirc embedding of a context, after looking a little more closely at what exactly the DisCoCirc context stores in the first place.

The DisCoCirc context

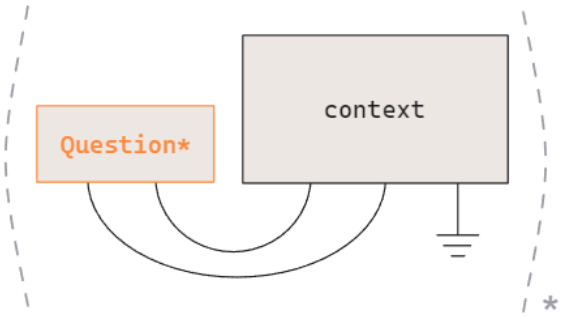
Given the noun-based presentation of DisCoCirc, a text is formalised as a process that describes how its nouns change. If we additionally specify their initial state, the resulting diagram is a representation of what the nouns are like at the end of the story. The natural kind of information we might want to extract is then about the nouns and their relationships: '*Is Mary in the kitchen?*', '*What colour is the chair?*'. These questions additionally presume a fixed moment in time (usually taken to be the end of the story), however we might also be interested in more sophisticated questions that query not just the final state, but the entire process that produced it: '*Was Mary in the bathroom before the kitchen?*'.

2.3.2 Implementations

We explore two ways for implementing questions, with a few variations each.



(a) DisCoCat-style question



(b) DisCoCirc-style question

Figure 2.8: Question types

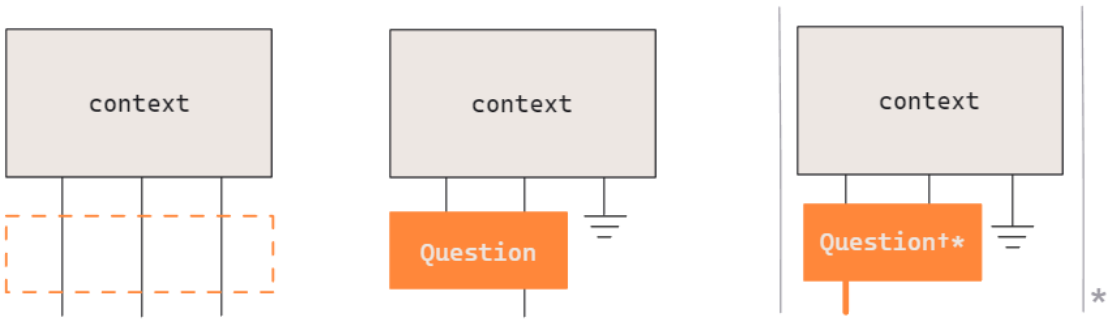
DisCoCat-style

We call questions framed as a way to extract information from a context directly DisCoCat-style, as the encoding of the question diagram is intended to compute the answer directly. The output type `S` of the entire sentence is then the space of possible answers to the question asked. The question diagram is connected to the story's DisCoCirc diagram using the usual rules for DisCoCirc. Figure 2.8a shows the basic DisCoCat-style question schema, while Figure 2.9 suggests some more examples.

DisCoCirc-style

A primitive notion in DisCoCirc is the idea of comparing texts. Although this is often presented using adjoints, it can be represented equivalently using cups to connect the outputs of a pair of contexts. The resulting number can be interpreted as measuring the overlap between the two texts, and acts as a similarity measure. Adapting this to questions with a known series of answers, we can construct a diagram for each possible answer and compute the overlap of each with the context. We can then take the answer with the highest overlap to be the model's output.

Unlike the DisCoCat-style questions, DisCoCirc-style questions require extra processing in order to generate the possible answers: for each answer, we generate a sentence capturing the state that would make that answer true (called an



(a) Id (b) Narrowing (c) Dense Coding

Figure 2.9: DisCoCat question examples (a) In some cases, the question may ask what the final state of the world is - since this is exactly what we take the context to represent, the question is simply the identity. (b) Alternatively, the question might require only part of the context, perhaps narrowed down (eg '*What colour is the cake, given that Alice is sad today?*'). In this case, we take the question to act as a projection onto the case required. Finally, (c) is a quantum-specific variant of the DisCoCat approach intended to allow for dense-coding approaches. Rather than using a single question, the question can be implemented as a series of questions, which each contribute to the final output bit string.

assertion). For example given a binary question '*Is Mary in the kitchen?*', the resulting assertions are '*Mary is in the kitchen*' and '*Mary is not in the kitchen*'. Figure 2.8b displays a DisCoCirc-style question in a compact closed category, whilst Figure 2.10 makes use of adjoints.

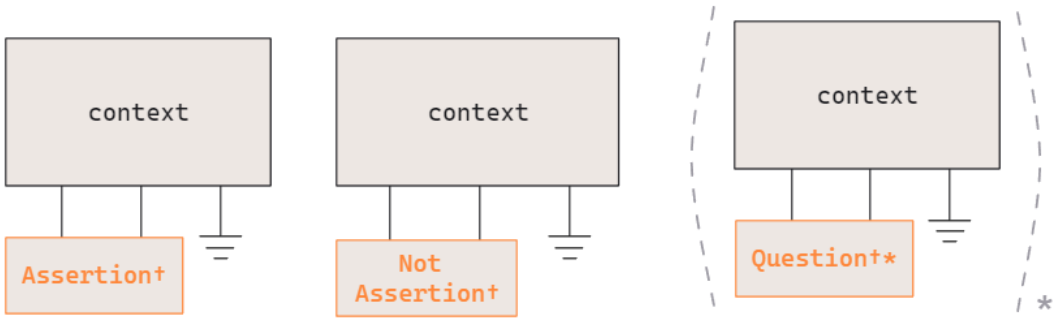
Sharing vocabulary

Although each word is assigned a single box, we might want to interpret words occurring in a question differently to words that occur in the main story. This would be in an attempt to capture the intuitive difference between *describing* and *querying* - questions are not intended to add to the description of the world, rather, they are meant to allow information to be extracted. Using a different kind of box for question words allows this distinction to be made explicit. For the DisCoCat-style models, this is only relevant when the output type is upgraded to be a series of nouns, such that it resembles DisCoCirc.

Requirements

Implementing questions introduces certain structural requirements for the syntax category of the model. Specifically, the category must be a discard-category so that nouns that aren't involved can be ignored, and be either compact closed or

have a notion of adjoints for DisCoCirc style questions.

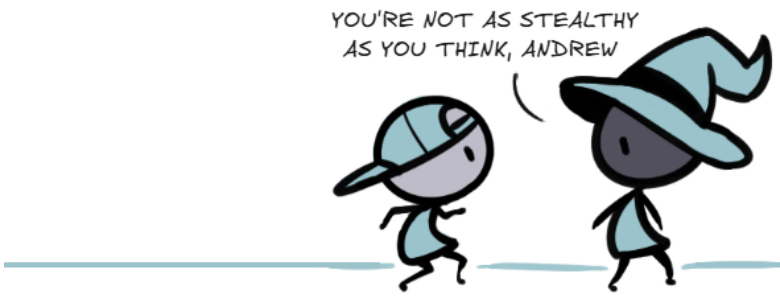


(a) Binary (b) Discrete-valued

Figure 2.10: DisCoCirc question examples. (a) Binary questions can be implemented as a pair of a positive and a negative assertion. (b) For questions whose answer can take discrete values, we can construct a list of (task-dependent) assertions to test individually. For example '*Where is Mary?*' might give rise to the sub-assertions '*Mary is in the kitchen.*', '*Mary is in the garden.*', '*Mary is in the library.*'.

Part II

Methodology



CHAPTER 3

Tasks and datasets

We have now established our basic model architecture, and what it means to be compositional. In this chapter will focus on the chosen reading comprehension tasks, analysing their characteristics, and in particular the ways in which they appear compositional. We consider two types of dataset: a synthetic dataset built for a proof-of-concept work (known as *following*), on which we measure productivity, and an extension of an existing synthetic dataset designed to test basic reading comprehension: *bAbI* (Weston et al., 2015; Tamari et al., 2021), specfically task 6, which allows for a full study of compositionality.

As with any ML task, the ultimate success of the model is highly dependent on the data it is trained on. With the ultimate goal of testing the model's understanding, we begin by studying some synthetic reading comprehension datasets, using binary questions to probe whether the model has understood a particular aspect of the context. In particular, we select tasks that exhibit compositional properties, with the goal of testing generalisation. Given the compositional nature of the tasks, we can then view successful compositional generalisation as evidence of an appropriate *understanding*, in that the model has learnt the correct underlying rules to solve the dataset.

3.1 Tasks

3.1.1 Following

We introduce the *following* dataset as part of a proof-of-concept work testing DisCoCirc's capability at productive generalisation (Duneau et al., 2024). The

	Following		bAbI task 6
1	*Andrew walks north.*	1	*Clara went to the bathroom.*
2	*Bill walks south.*	2	*Eric travelled to the garden.*
3	*Andrew follows Bill.*	3	*Eric picked up the football.*
4	*Bill turns around.*	4	*Clara moved back to the office.*
Q	*Is Andrew going in the same direction as Bill?*	Q	*Is Eric in the bathroom?*

Table 3.1: Example stories from the following and bAbI task 6 datasets.

stories considered consist of actors moving around in one of either two or four directions. At the start of each story, each actor is initialised to face one of the cardinal directions, after which they may either turn around by themselves or interact with another actor by following them, or going in the opposite direction. For each story, we pick two actors, and ask whether they are facing the same direction at the end of the story. An example story is shown in Table 3.1. In this preliminary work, many simplifications were made to allow the focus to be placed on the productivity of the model.

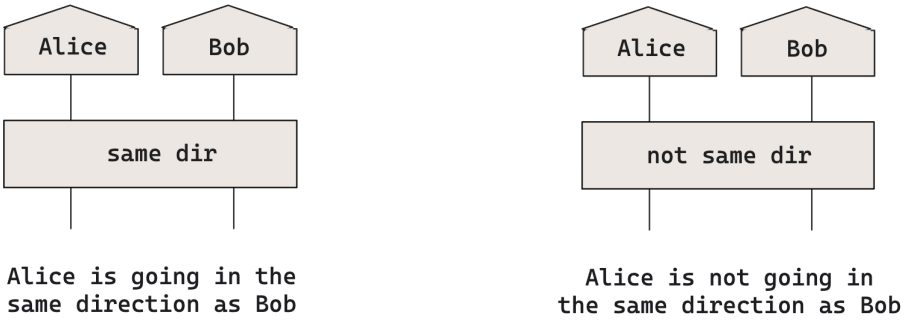
Vocabulary

Shape	Two-directional vocabulary	Four-directional vocabulary
Noun	*Andrew*, *Bill*, *Clara*, *Denise*, *Eric*, *Fred*, *Gillian*, *Heidi*...	
ITV	*walks north*, *walks south*, *turns around*	two-directional vocabulary, plus: *walks east*, *walks west*, *turns right*, *turns left*
TV	*follows*, *goes in the opposite direction of*, *goes in the same direction as*, *does not go in the same direction as*	

Table 3.2: Vocabulary present in the following tasks, by (simplified) DisCoCirc shape. Words exclusively used in the questions are underlined.

The grammar of the text inputs was artificially simplified by collapsing certain phrases and treating them as single words. Table 3.2 summarises the phrases used and their assigned part-of-speech for DisCoCirc parsing. We identify a two-directional restriction of the four-directional task, providing a simpler case study for comparison. The question decomposition as assertions is shown in Figure 3.1.

‘*Is Alice going in the same direction as Bob?*’

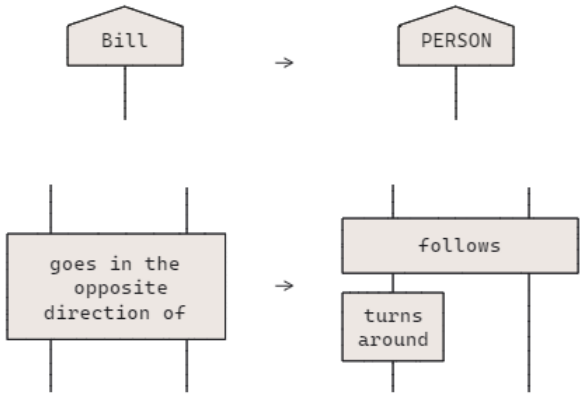


(a) (b)

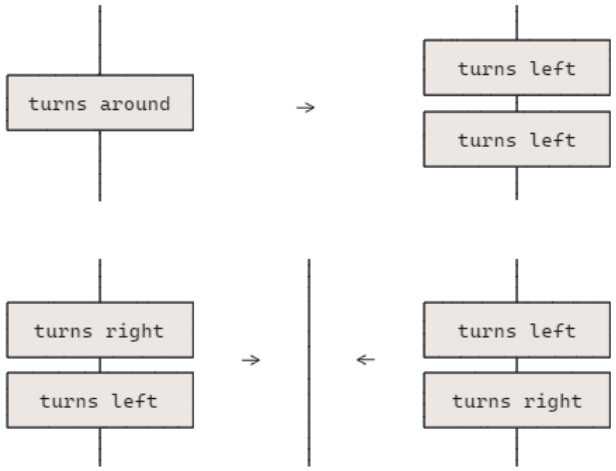
Figure 3.1: Assertions for the questions in the following datasets, as DisCoCirc diagrams. The binary question ‘*Is Alice going in the same direction as Bob?*’ is decomposed into a pair of assertions covering the (a) positive, and (b) negative answer cases. The would-be frame for the negated question is collapsed using the pancake method of Section 2.1.5.

Rewrites

We make extensive use of built-in semantic rewrites, as detailed in Figure 3.2. There are no frames to expand after the vocabulary simplification (in some cases, the simplifications act like a pancake frame expansion).



(a) Two-directional



(b) Four-directional

Figure 3.2: Rewrite rules for the *following* dataset: (a) All names are replaced with a generic `PERSON` state. We also express *goes in the opposite direction of* in terms of `follows` and `turns around`. (b) The four-directional dataset includes additional rewrites concerning the quarter-turns: `turns around` is equal to `turns left` twice in a row, and `turns left` and `turns right` are taken to be inverses.

3.1.2 bAbI 6

The bAbI tasks are intended to cover a range of basic inference and reading comprehension prerequisites. We chose to focus on task 6 in particular, as the format was suited to the DisCoCirc framework, whilst also containing sufficiently interesting structure for testing compositionality.

These stories describe actors moving around a series of rooms. Additionally, the actors may either pick up or discard items. For task 6, the position of the items is irrelevant to the task, as the questions are always of the form '*Is person x in place y?*'. An example story is given in Table 3.1.

Vocabulary

For the bAbI dataset, we did not simplify the vocabulary. We thus obtain higher-order boxes of various shapes on top of the basic types encountered in the following dataset. Table 3.3 displays the vocabulary while Figure 3.3 shows a typical question and associated assertions.

Shape	**Vocabulary**
Noun	*Andrew*, *Bill*, *Clara*, *Denise*, *Eric*, *Fred*, *Gillian*, *Heidi* *apple*, *football*, *milk*, *slippers* *kitchen*, *office*, *hallway*, *bedroom*, *garden*, *bathroom*, *cinema*, *park*
ITV	*travelled*, *moved*, *journeyed*, *went*, *in*
TV	*discarded*, *dropped*, *left*, *put*, *grabbed*, *picked*, *took*, *got*
HO (1, 1)	*back*
HO (2, 1)	*to*, *is*
HO (2, 2)	*up*, *down*, *not*

Table 3.3: Vocabulary present in the bAbI 6 task, by DisCoCirc shape. Words exclusively used in the questions are underlined.

Rewrites

For this dataset, we only use the basic grammatical rewrite for *the*, converting it to an identity. Further rewrites such as identifying synonyms in the verbs and grouping the nouns together would trivialise the systematicity and substitutivity compositionality tests, so we did not apply them here. Frames were expanded using the sandwich functor, which does not interfere with localism on the fragment of language involved.

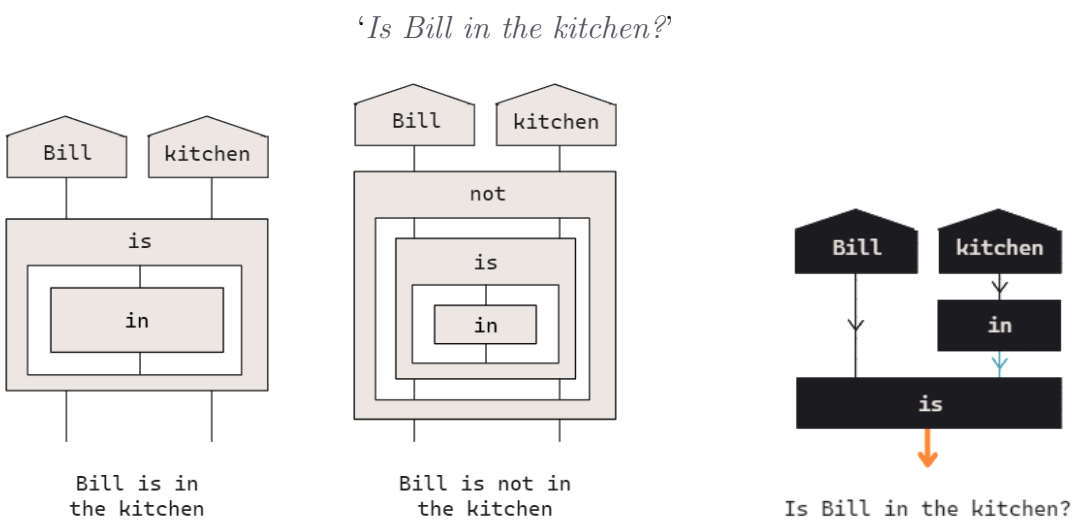


Figure 3.3: (a) bAbI 6 assertions as DisCoCirc diagrams. The binary question '*Is Bill in the kitchen?*' is decomposed into a pair of assertions covering the two possible answer cases. (b) The question as a circify-ed DisCoCat diagram.

3.2 Analysis and characterisation

For each dataset, we collect metadata about each entry to inform our analysis of the results. These come under three main sections: *compositionality* measures capture the extent to which a certain aspect of compositionality is being tested by the given datapoint; *interpretability* measures are used to inform our analysis of the trained models; and *resource* measures quantify the expected difficulty of evaluating a given datapoint.

3.2.1 Compositionality

For each of the compositionality tests, we devise measures intended to track the difficulty of each datapoint with respect to the given test. We note that many of the tests require an initial 'base' subset to be chosen, from which the rest of the dataset is generated. This is specified for each measure.

For training purposes, we introduce an extra metric to track the training group each datapoint belongs to. This is calculated as a combination of the compositionality and balancing measures, depending on the particular task.

Stratum The training group assigned to this datapoint.

Productivity

In this case, we take as source the set of single sentence texts that can be formed from our vocabulary. We also fix the possible sentence structures, such that this subset is combinatorially bound by the vocabulary. As such each application of productivity will append a text to another, so the text depth is a natural measure of productive difficulty.

Text depth	The number of sentences in the context.
Noun width	The number of distinct nouns (such as *Bob*, *Bathroom*, *Basketball*) occurring in the context. This is generally correlated with context depth.

Systematicity

Here, we pick a subset of the possible pairs of nouns that can occur as arguments to the same verb, as a base set. We pick four people and four locations, leaving the objects independent. We can then derive the systematic difficulty by tracking the number of distinct pairs not from this base set involved in a given context.

Noun groups	The subset of the nouns that this context is restricted to. We specify the groups per class of noun (ie people, locations and objects) as there are no possible *intra*-group interactions.
Distinct pairs	The number of distinct noun pairings encountered in the text. We track both the total number, as well as the number relative to the base subset.

Substitutivity

To assess substitutivity, we partition the data into equivalence classes, picking a single representative from each to form our base set. The substitutive difficulty is then measured as the number of substitutions that must be made to convert the text into the canonical representative of its class. The representative member for each class is picked randomly to maintain an approximately even distribution of vocabulary over the base set.

Structure The story structure captures the basic shape of the story as a directed graph derived from the DisCoCirc diagram, where box names are grouped by synonyms. Thus, stories with the same structure are equivalent up to their synonyms. This also admits consistent permutation of nouns that share the same type.

Synonym distance For stories with the same structure, we count the number of synonym replacements needed to get from one story to the representative member of the equivalence class.

Overgeneralisation

For this aspect of compositionality, we first fix a set of data as our ground truth, and from there generate an entirely corrupted dataset. The levels of overgeneralisation we assess will then depend on the probability that controls which dataset (pure or corrupted) each datapoint is sampled from. As our task is binary, we corrupt datapoints simply by flipping their final labels.

We leave more sophisticated corruption methods for future work. For example, 'systematic corruption' where the meaning of a certain box is altered in a context-dependent way (for example, by replacing `follows` with `follows*`, which applies normally to everyone except Alice), or probabilistically removing or changing an entire (relevant) sentence from the context without changing the final label.

Corrupt This tracks whether the dataset entry has been artificially corrupted.

3.2.2 Interpretability

Additionally, we track some further task-specific details to help interpret and contextualise model performance. Balancing metrics are used to ensure the dataset distribution remains evenly spread, to help reduce model bias. Difficulty measures intend to capture aspects of a story that can make it more difficult to solve than others. A more detailed break-down of the solution is also tracked.

Balancing

Answer class The expected answer class for the question (usually either *yes* or *no*).

Difficulty

Support depth	The number of sentences between the last relevant statement and the end of the context. This is intended to help track the effect of 'scrambling'.
Inference steps	The number of sentences required to answer the question. For bAbI 6, this is always 1 by design, whilst the *following* dataset includes instances with more steps.
Confoundings	A confounding is a sentence involving the nouns in the question that does not contribute to the final answer. We track the confoundings affecting the nouns involved in the question separately.
Density	The density of a text is measured as the ratio of multi-noun boxes to single-noun boxes. Specific to the *following* dataset, this captures how much interaction occurs between characters, and is also correlated with the (quantum) resource costs.

Solution

Question nouns	The nouns involved in the question.
Target solution	The solution that provides the correct answer to the question. In the bAbI task, this would be the actual location of the person in the question, while in the *following* case it is the respective final directions the people in the question are facing.
Separable	*Following* only: whether the directions faced by the actors in the question are causally linked.

3.2.3 Resources

Finally, we also measure some further metrics once the datasets have been fully compiled, to track the resource complexity of evaluating the datapoints.

General

Flops	The number of flops required to evaluate the datapoint as a tensor network.
Memory size	The maximum intermediate size of tensor that must be stored whilst evaluating the datapoint as a tensor network.

Quantum

Two qubit gate depth	The two-qubit gate counts, both pre- and post-compilation, for a specific target device.
Qubit count	The total number of qubits required to evaluate the datapoint. Pre-compilation will in general *not* assume qubit reuse, whilst post-compilation can significantly reduce the required number by allowing it - it is therefore important to track both to get a more representative idea of the required resources.

3.3 Datasets

Here we visualise the characteristics of the datasets used in this work.

3.3.1 Following

For both the two- and four-directional tasks, we generate five sub-datasets of varying density. The density of a story is the number of two-actor interactions within the story divided by the number of sentences in the story. Generally, the denser the dataset, the more two-actor interactions present, and the more difficult the corresponding quantum circuit will be to evaluate. Table 3.4 summarises this.

Dataset splits

The dataset generation was controlled according to noun width, context depth, and answer class. Figure 3.5 shows how the resources required to evaluate a particular dataset entry scale as the noun-width increases, which in turn informed our decisions for splitting the dataset. We based the dataset splits on the number of nouns present in the text, which is correlated with the context depth. Table 3.5 shows a detailed overview of the splits used in our experiments, visualised in

Dataset	Two-directional		Four-directional	
(% density)	*Train, Validation, Valid Comp*	*Test*	*Train, Validation, Valid Comp*	*Test*
simple	26.5	24.8	26.6	25.0
deeper	48.2	54.3	48.0	54.3
less-dense	50.0	66.5	50.4	66.6
dense	58.2	71.6	58.1	71.6
superdense	68.7	77.5	68.7	77.5

Table 3.4: Average densities of the two-directional and four-directional datasets.

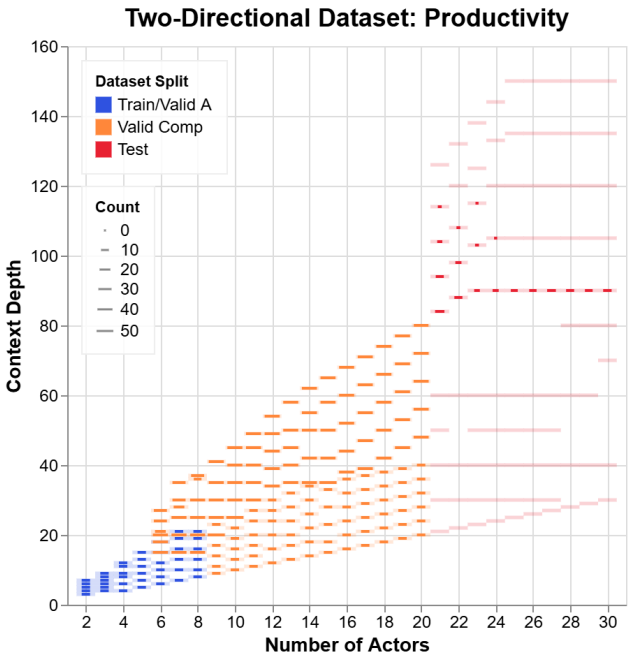
Figure 3.4. In particular, although we generated stories with up to 30 nouns and 150 sentences, we were not able to evaluate all of these with the resources available. Figure 3.6 displays the distribution of datapoints from the dense and superdense *Test* datasets that compiled down to 20 qubits.

Two-directional dataset The model is trained on the simple dataset stories with up to 8 actors. 20% of the simple dataset with up to 8 actors is used as validation data; we refer to this set as *Validation*. The stories from all other (deeper to superdense) datasets with up to 8 actors, and the stories from all datasets (simple to superdense) with 9 to 20 actors, are used as a second compositionality validation dataset, called *Valid Comp*. This set is used to pick the model with the best generalisation performance. Stories from all datasets with 21 to 30 actors, which compile down to at most 20 qubits, form the test set. We simulate up to 20 actors and send a selection of circuits between 9 and 30 actors to Quantinuum's ion-trap quantum computer H1-1 (these sets are called *Valid Comp (H1-1)* and *Test (H1-1)* depending on the dataset the datapoints were sampled from).

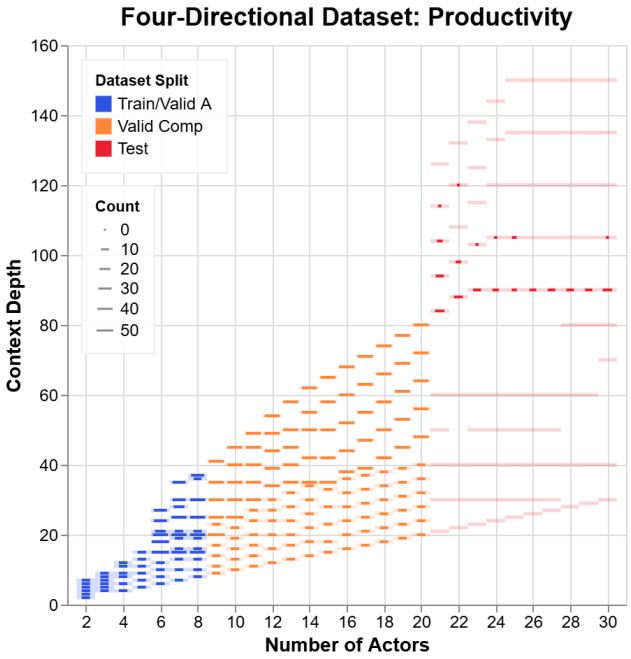
Four-directional dataset The model is trained on all (simple to superdense) stories with up to 8 actors. 20% of the dataset with up to 8 actors is used as validation data. The stories from all datasets (simple to superdense) with 9 to 20 actors are used as a second compositionality validation dataset (*Valid Comp*), which is used to pick the model with the best generalisation performance. Stories from all datasets with 21 to 30 actors, which compile down to at most 20 qubits form the test set. We select a subset of these circuits to evaluate using shot-based noisy simulation, and call these *Test (qujax)*.

Dataset	Two-directional			Four-directional		
	Samples	Actors	Density	Samples	Actors	Density
Train	80%	2-8	simple	80%	2-8	all
Validation	20%	2-8	simple	20%	2-8	all
Valid Comp	all	9-20	simple	all	9-20	all
	all	6-20	deeper-spdense			
Valid Comp (H1-1)	120	9-20	dense		-	
	120	9-20	spdense			
Test	all	21-30	all	all	21 - 30	all
Test (H1-1)	82	21-30	dense		-	
	47	21-30	spdense			
Test (qujax)		-		86	21-30	dense
				45	21-30	spdense

Table 3.5: Detailed splits used in our experiments. A visual representation is provided in Figure 3.4. Note that superdense is abbreviated as spdense.

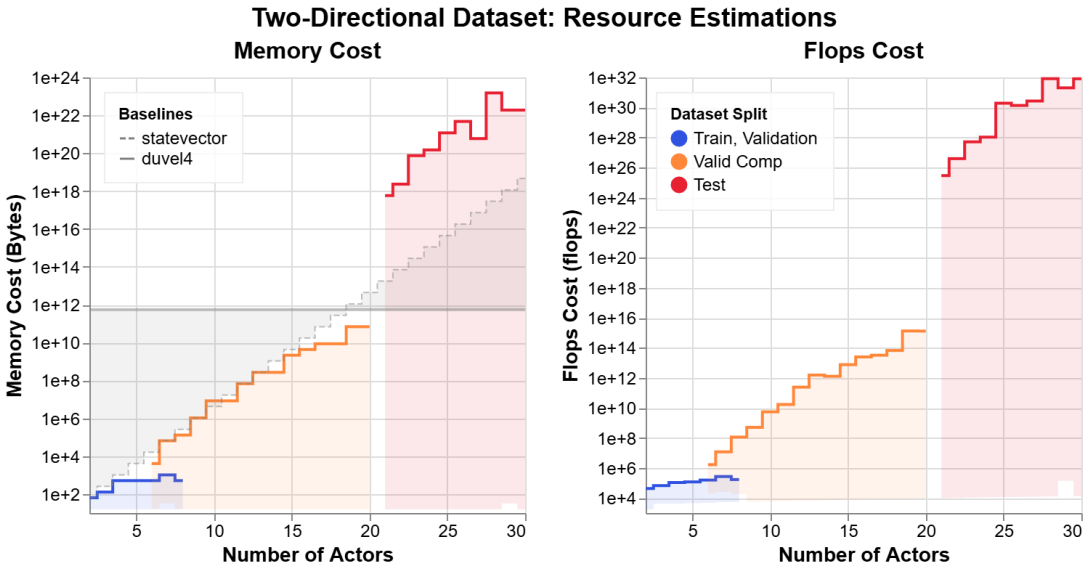


(a) Two-directional

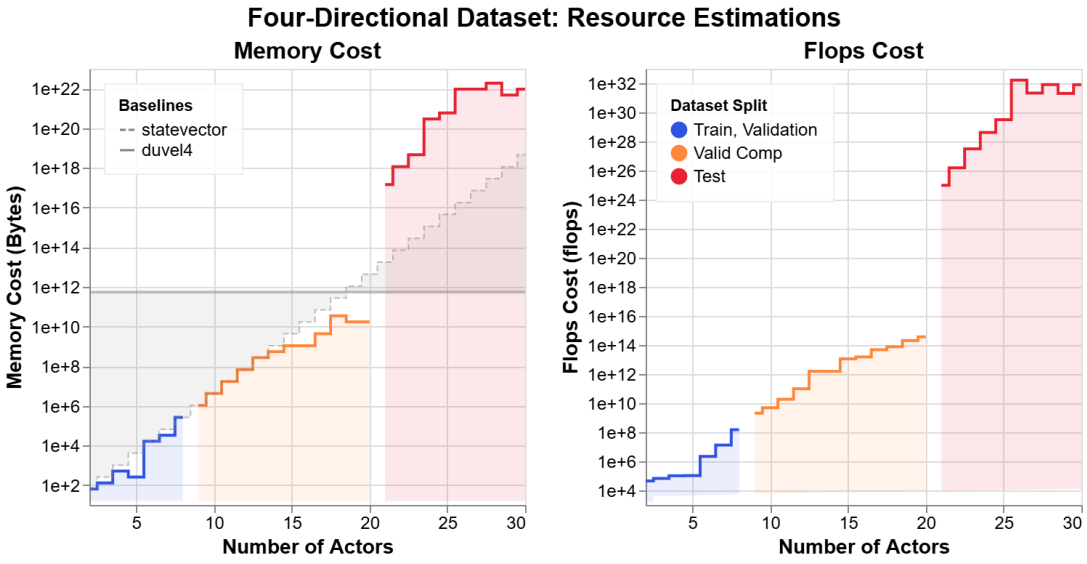


(b) Four-directional

Figure 3.4: Visualising the dataset productivity measures, for combined densities over the (a) two-directional and (b) four-directional datasets. The lightly shaded regions correspond to points for which data was generated. Not all of these were evaluated (from the *Test* dataset in particular). The solid dashes mark which story shape combinations were sampled, with the width of the dash representing the number of samples present. Notice that the selected *Test* dataset is restricted to shapes that are strictly larger than any selected for *Train* or *Valid Comp*, starting from 21 nouns and 81 sentences.

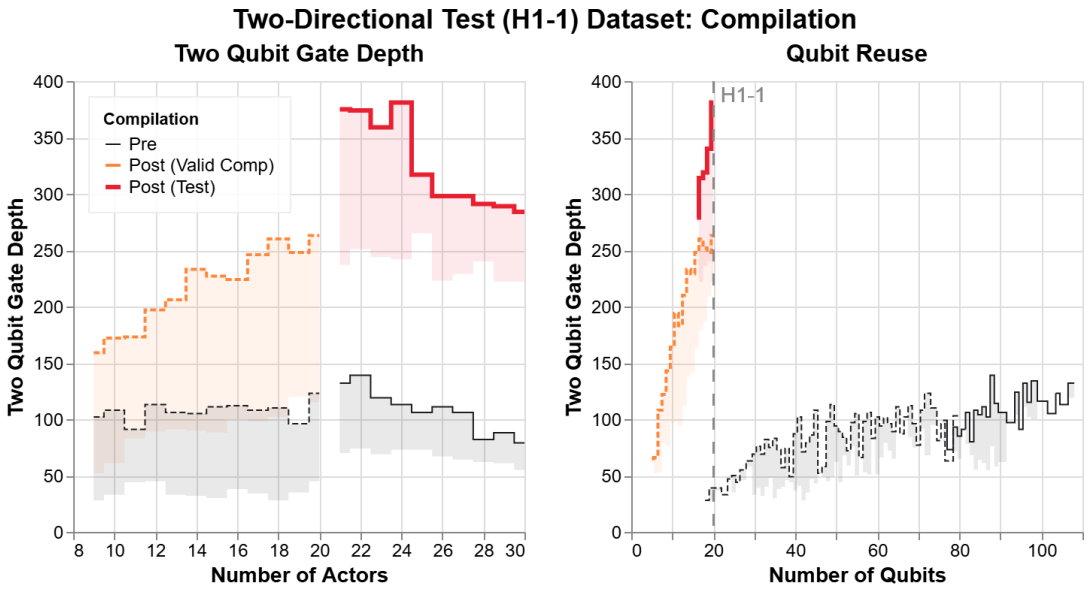


(a) Two-directional

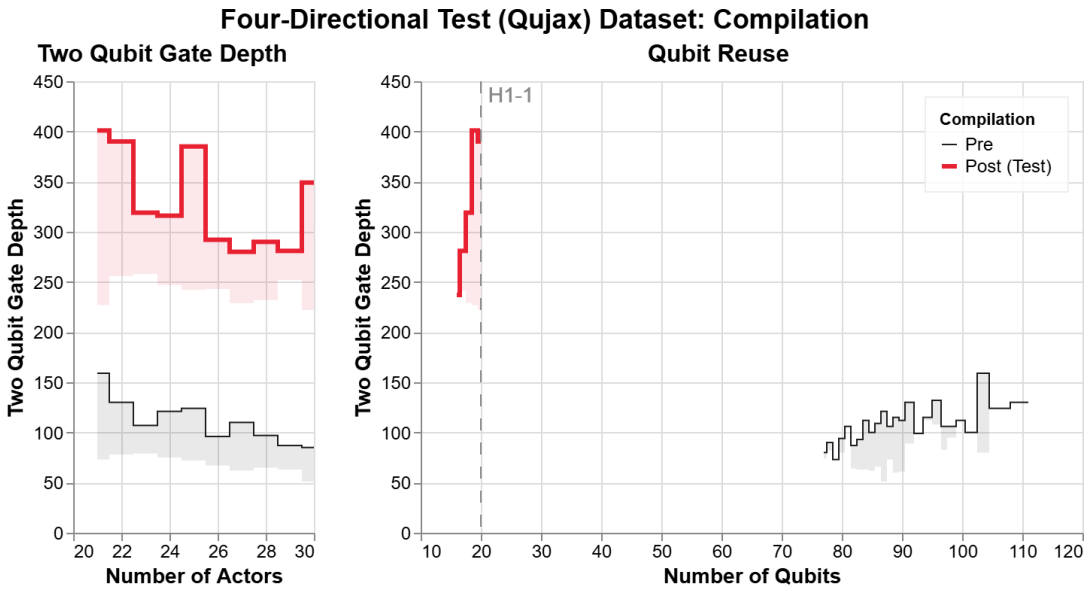


(b) Four-directional

Figure 3.5: Estimated resources required to evaluate the quantum circuits for each datapoint in the (a) two-directional and (b) four-directional datasets. We plot the upper bound for each dataset split as a line, with shaded area showing the extent of the datapoints. The *duvel4* baselines on the memory cost plots represent the maximum memory available on the device used to train the models, whilst the dashed grey statevector baseline is a lower bound obtained by evaluating the quantum circuits as a statevector, assuming usage of a single ancilla. The difference between the statevector and duvel baselines is shaded to highlight it.



(a) Two-directional



(b) Four-directional

Figure 3.6: Effect of compilation and qubit reuse on the quantum circuits used to evaluate the (a) two-directional and (b) four-directional datapoints for the *Test (H1-1)* and *Test (Qujax)* datasets. For the two directional dataset, we also show the *Valid Comp (H1-1)* dataset. In each case, the pre-compilation values are plotted in grey with a thin line highlighting the maximum. The post-compilation results are plotted in colour with a thicker line. The *Valid Comp (H1-1)* entries are shown dashed to help distinguish them. The restricted *Valid Comp (H1-1)*, *Test (H1-1)* and *Test(Qujax)* are sampled from the datapoints in *Valid Comp* and *Test* that compiled down to at most 20 qubits.

3.3.2 bAbI 6

We generate a dataset for each aspect of compositionality.

Productivity

The productivity dataset matches the original bAbI 6 data most closely, so we will take this dataset as a base from which to derive others where needed. The dataset is generated procedurally by sampling sentences one at a time until a desired depth is reached. We then check that the candidate entry matches any further selection criteria, and if so add it to the dataset. For productivity, we enforced an even split between answer classes, and an even spread of context depth and noun width. We also controlled the support depth to ensure there were sufficient datapoints present at the higher depths, however as the support depth is constrained by the context depth, there are necessarily more entries with a lower depth. We generate two datasets drawn from the same distribution that we split as *Productivity* and *Productivity'*[1]. We visualise the strata splits and balancing in Figure 3.9, Figure 3.7 and Figure 3.8.

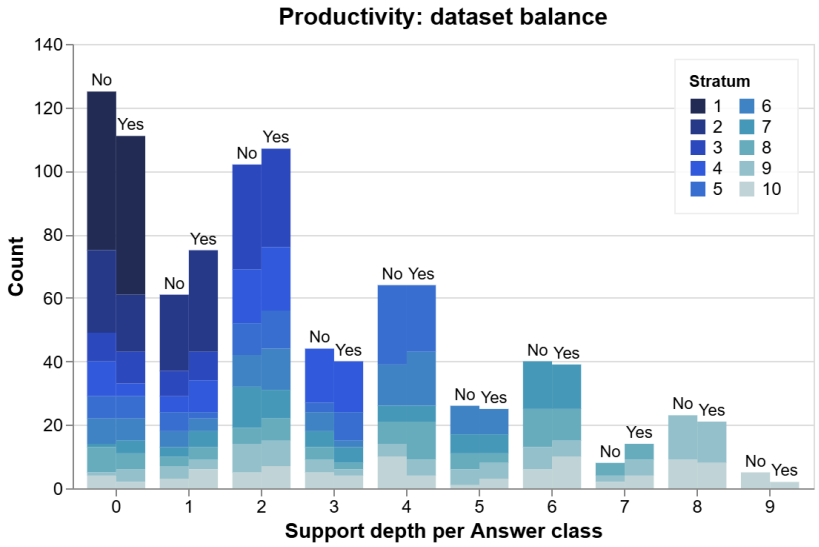


Figure 3.7: Visualising the balancing for the primary *Productivity* dataset. The stratum in this case is the context depth for each story, represented by colour. The distribution of context depth is plotted according to the support depth and answer class.

[1]In most situations, we shall sample both the *Train* and *Test* sets from the *Productivity* dataset, whilst *Productivity'* is only used as an additional evaluation set to validate model performance on unseen elements drawn from a similar distribution to the training set.

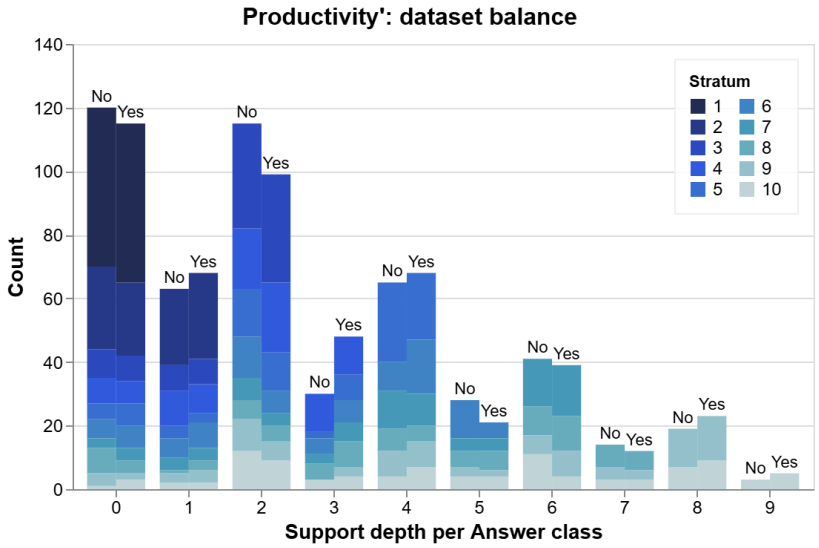
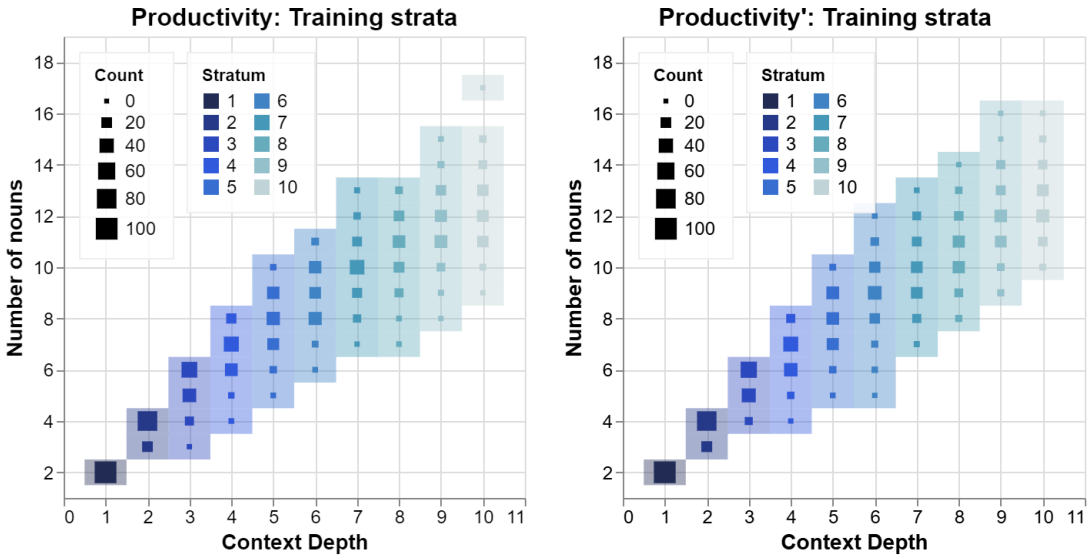


Figure 3.8: Visualising the balancing for the secondary *Productivity'* dataset. The stratum in this case is the context depth for each story, represented by colour. The distribution of context depth is plotted according to the support depth and answer class.



(a) (b)

Figure 3.9: Visualising the distribution of story size, according to the context depth and number of nouns, for the (a) *Productivity* and (b) *Productivity'* datasets. The stratum in this case is the context depth for each story, represented by colour. In each case the region covered by each stratum is shaded, with the size of the points corresponding to the number of datapoints sampled for that shape combination.

Systematicity

When generating the systematicity dataset, we split the nouns into sub-groups (detailed in Figure 3.10), then generated entries limiting the nouns involved in each group. Similarly to productivity, we balanced the generated data according to context depth and answer class (Figure 3.11). We can then visualise the systematicity splits by inspecting which of the person and location sub-groups were paired by each entry, as shown in Figure 3.12. The overall dataset is slightly biased towards *no*, as a consequence of the base case, in which the base group isolates a single person and location to be paired with the others. For a single person (respectively location), there is only one way for them to be in the location, while there are six ways for them not to be. We chose to prefer including more examples of systematicity in this case than to enforce an exact balance in answer class. The resulting baseline accuracy is 51.6%, rather than the expected 50%.

Notice that there is some overlap between the noun groups in Figure 3.10, such that a given datapoint may fall into multiple groups. In such cases, we assign the most restrictive group available. For example, a datapoint containing *Bill* and *Clara* will fall into groups 3, 4 and -2, but will be assigned to group 3 as this is the smallest group.

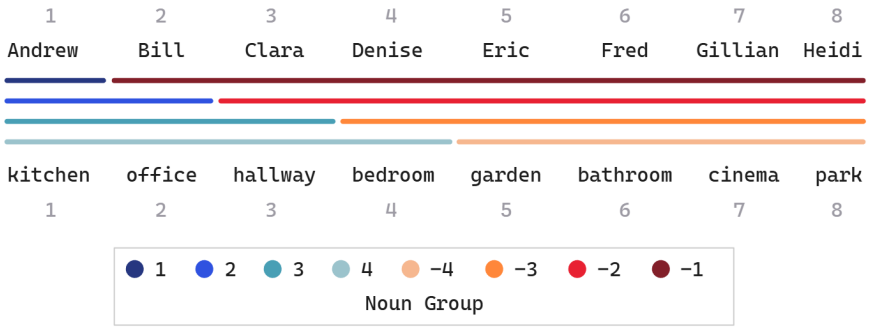
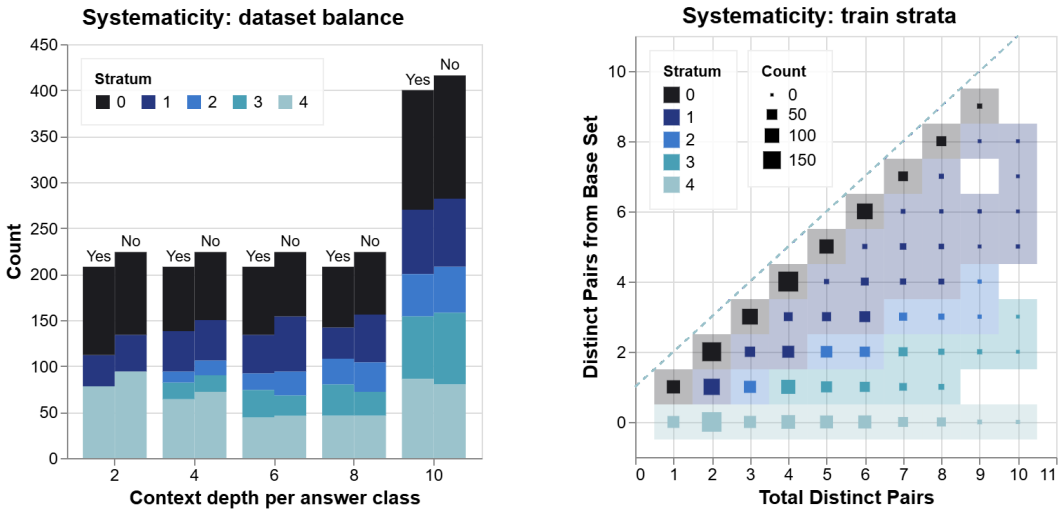
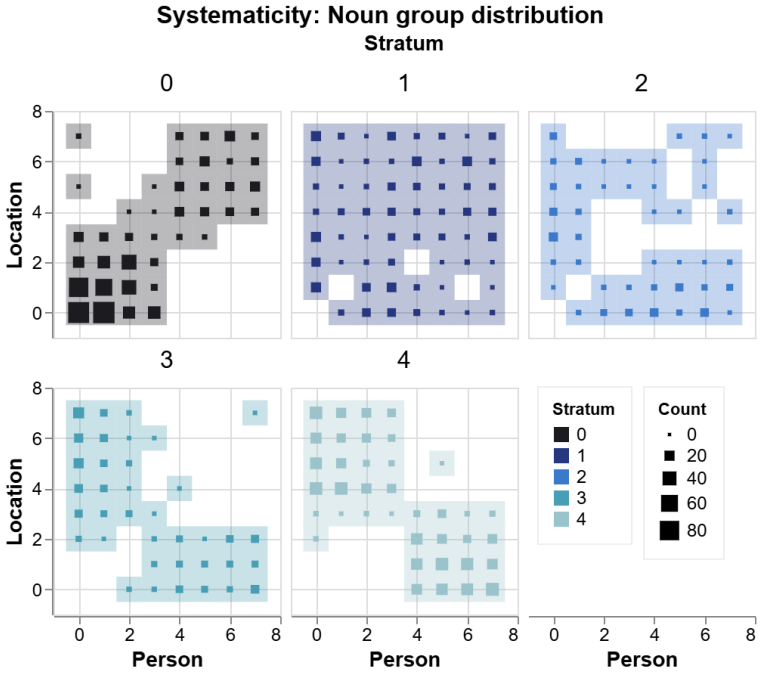


Figure 3.10: The noun groups used for generating the systematicity dataset. The top and bottom layers identify the people and the locations respectively. The order of the nouns is kept fixed so they can be indexed by number (shown in grey). The groups are coloured, with positive indices coloured in blue and their complements (negative indices) coloured in orange.

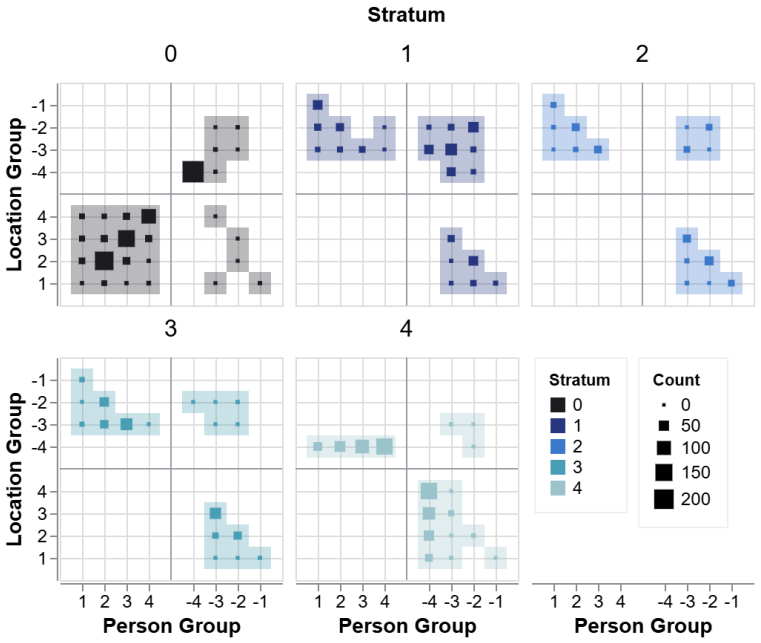


(a) (b)

Figure 3.11: Visualising the splits for the systematicity dataset. The object interactions were not considered when counting the systematicity pairs, so some stories may contain nouns from groups outside the recorded noun group. (a) We plot the stratum balancing across context depth and answer class. Notice that for a context depth of 2, not all strata are available. The bars show the cumulative number of points for each depth. The stratum for each datapoint is calculated as a function of the total distinct pairs present, and the distinct pairs that occur in the base set. We visualise this relationship in (b). The regions belonging to each stratum are coloured accordingly. The size of the point plotted in each region captures the number of datapoints present. The dashed line represents the limit where all pairs seen belong to the base set: these examples form the easiest stratum, whilst datapoints on the bottom row contain no pairings from the base set.



(a)



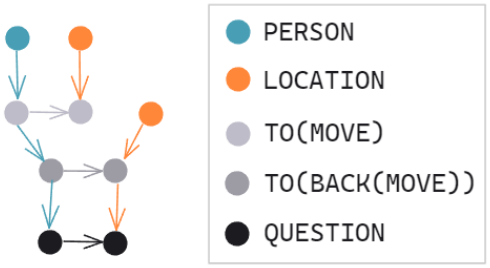
(b)

Figure 3.12: Visualising the distribution of nouns present in each training stratum for the systematicity dataset. (a) plots which nouns are seen together in the questions, whilst (b) plots which noun groups are present throughout the context, excluding the question. In each case, we shade the region covered by each stratum. The size of the points plotted for each person and location (group) combination represents the number of datapoints available.

Substitutivity

The substitutivity dataset was initially derived from the base productivity dataset. For each datapoint, we identified its structure such as that in Figure 3.13. The structure of each datapoint abstracts away both the particular choice of synonym as well as some symmetries in the ordering of sentences that DisCoCirc diagrams contain by virtue of being symmetric monoidal diagrams. We then sampled 500 unique structures, balanced across context depth and answer class to form our dataset. For each sampled structure, we first construct the base example by fixing an element of the vocabulary for each verb class. We then generate four further entries from this structure, with varying synonym distances.

These are then split into tiers according to the synonym distance exhibited by the datapoint. In order to maintain somewhat balanced tiers, we consider both the absolute number of synonym replacements needed, as well as the number of replacements for movement type verbs only, as these are the ones that provide relevant information for the final answer. The splits are visualised in Figure 3.14.



(a) A structure representation

1. *Clara journeyed to the park.*
2. *Clara went back to the garden.*

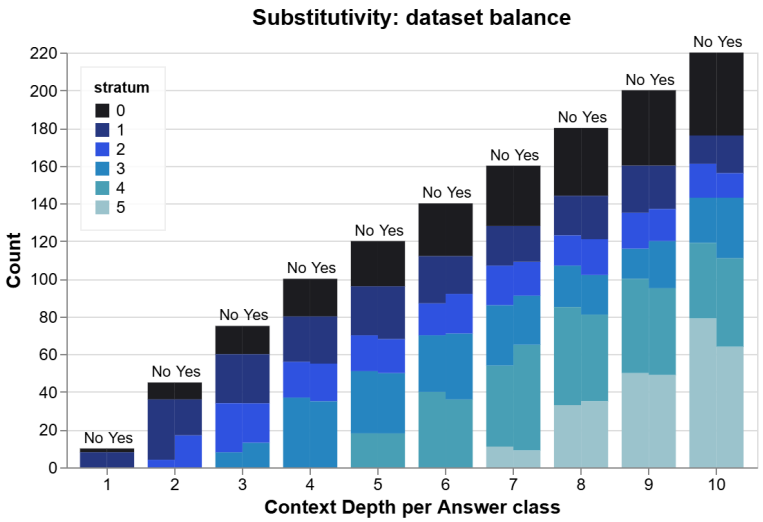
Q. *Is Clara in the garden?*

1. *Bill travelled to the kitchen.*
2. *Bill moved back to the office.*

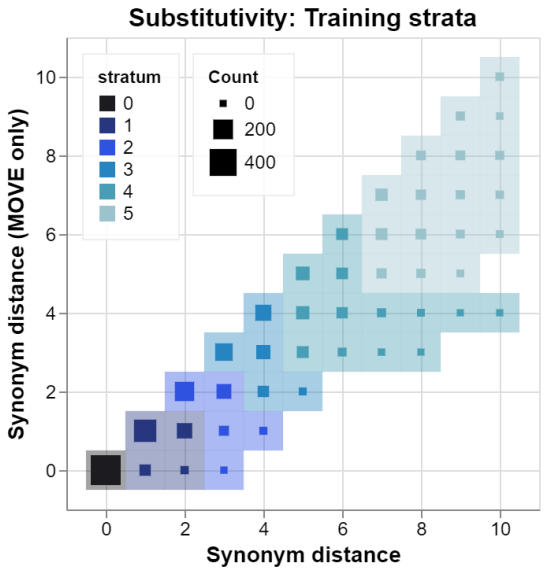
Q. *Is Bill in the office?*

(b) Compatible stories.

Figure 3.13: A representation of a structure as a coloured directed acyclic graph. The nouns chosen for the question are included in the structure to fix the answer class. Boxes are represented as a sequence of nodes in order to distinguish the subject and object.



(a)



(b)

Figure 3.14: Visualising the substitutivity dataset. (a) Visualising the balance of splits for the substitutivity dataset structures according to the context depth and answer class. The splits are not exactly balanced per stratum due to the randomness present in data generation. The stratum is necessarily somewhat correlated with the context depth, as the number of synonyms available for replacement depends on the number of sentences in the story. (b) Visualising training strata in terms of synonym distances. The region covered by each stratum is shaded, while the size of the point repesents the number of samples available.

Overgeneralisation

For overgeneralisation, we take the productivity dataset as the uncorrupted dataset. We then generate some further datasets with $10\%, 20\%, 30\%$ and 50% corruption. Note that all corrupted datapoints in the 10% dataset are also corrupted in the 20% dataset, and so on. The datapoints to be corrupted were sampled per answer class and context depth to avoid biasing the dataset, as shown in Figure 3.15.

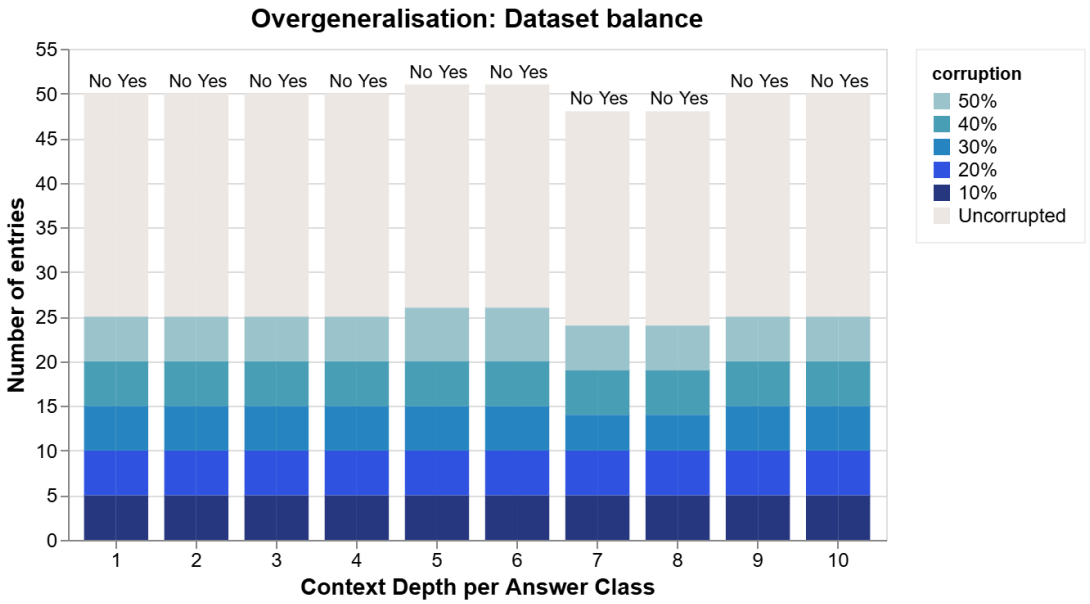
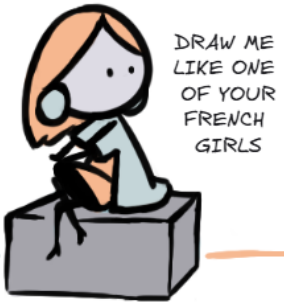


Figure 3.15: Visualising the splits for the Overgeneralisation datasets according to the context depth and answer class. The percentage of corrupted datapoints is represented by the colour of the bar, and is cumulative.



CHAPTER 4

Models

Recall that a DisCoCirc model implements a semantic functor. We will consider two classes of model, which differ primarily in the choice of monoidal product. Standard presentations assume a monoidal product that isn't cartesian - such as the tensor product $\otimes$ over vector spaces. We will also investigate the option of using a cartesian product $\oplus$, and compare both approaches. We call the first quantum and the second classical after the specific semantic categories we chose to investigate - namely quantum circuits and neural networks.

While the DisCoCirc formalism places some requirements on the semantic category we can chose, we must also contend with practical requirements in order for the model to be implementable. We consider two approaches - quantum and classical - that capture the theoretical tradeoff to be made between the choice of monoidal product.

4.1 Quantum

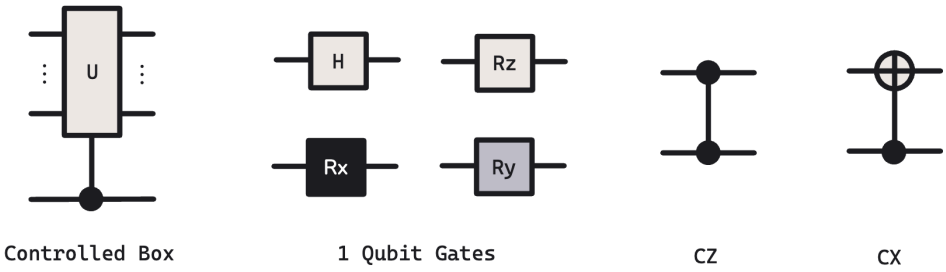
Having selected $\otimes$ as our monoidal product, an obvious choice of semantic category is $FdHilb$, the category of finite dimensional Hilbert spaces and linear maps. However, diagrams expressed in this category correspond to general tensor networks, and are therefore hard to evaluate in general. We hence chose to compromise some degree of expressibility and constrain our models to produce quantum circuits, in $CPM(Hilb_2)$.[1] Diagrams in $CPM(Hilb_2)$ are drawn using thick wires, and read

[1] See Coecke and Kissinger (2017) for details on the CPM construction. In brief, this involves 'doubling' diagrams in $Hilb_2$ - a restriction of $FdHilb$, in which all objects decompose as a tensor product of the two-dimensional Hilbert space $\mathcal{H}_2$ - such that they now capture positive operators and density matrices rather than complex vectors and linear maps.

from left to right. This is a symmetric, closed, discard category with a braiding, cups, caps and a discard. For accessibility, we use the conventional quantum circuit diagram representations of the boxes rather than their ZX-calculus shapes. A quick reference is given in Figure 4.1.



(a)



(b)

Figure 4.1: Generators for $CPM(FdHilb_2)$. Diagrams are read left to right. (a) Objects and morphisms. Each wire corresponds to a single qubit. Thanks to the Stinespring dilation Coecke and Kissinger (2017)[Theorem 5.61], we can write a causal quantum map as a unitary followed by some discards. Note that unitaries have the same number of input as output wires. (b) Special box notation for common gates: a controlled gate CU implements the unitary U if the control bit (marked with a circle) is in the state $|1\rangle$, and does nothing when it is in state $|0\rangle$. We shade certain single qubit gates differently to make them easy to distinguish. Finally, the controlled Z and X gates are given special notations, as shown.

4.1.1 Constraints

This in turn poses some further practical constraints:

Avoid post-selections

Post-selections typically incur an exponential cost, so our implementation will ideally impose a bound on the number of post-selections we might encounter.

Generally, we will constrain ourselves to tasks that require only local measurements to solve, such that the number of post-selections required will grow with the size of the question rather than the size of the context.

Avoid extra ancillas

Introducing and removing wires partway through the circuit incurs a cost - we might therefore chose to maintain the number of wires in the circuit as constant as possible. States, effects and frames are the only boxes that allow variation on the number of wires present on any given slice of the text circuit. We assume states are all pulled to the start of the circuit, whilst effects occur only as daggered states in the question. Frames are hence the only source of variation, which we remove by expanding them using the sandwich method.

Discard the ancillas

If we are to include ancillas to increase the expressivity of certain boxes, the natural choice when a wire is to be removed is to use a discard, since post-selections are to be avoided. The resulting boxes may then be implemented as mixed channels rather than just unitaries. This in particular allows for projections along a subspace, but notably not projections onto a single (specified) outcome.

Ansatze choices

Though a universal gateset can in principle implement any unitary, such a circuit is not necessarily efficient in terms of gate-count. Rather than learning an arbitrary tensor, we hence impose that each box implements a circuit of a certain shape, known as an ansatz (plural ansatze). The choice of ansatz affects the expressivity of the circuit - some are more expressive than others, often at the cost of a deeper gate count. Sim et al. (2019) present and analyse a selection of ansatze.

In preliminary work, we considered the Instantaneous Quantum Circuit (IQP) ansatz, and circuits 4, 9 and 14 (Sim4, Sim9, Sim14) from Sim et al. (2019), as well as a variation of Sim9 with the Rz gates replaced with Rx (Sim9Rx), with a variety of layers (from 1 to 3). These are illustrated in Figure 4.2 . We ultimately chose the Sim4 ansatz, with three layers, as this provides fairly high expressivity that scales well with the number of qubits in the ansatz. Additionally, the gates involved aligned well with the gateset of Quantinuum's H1-1 ion trap quantum computer (Quantinuum, 2023), which we used to implement the two-directional following model (see Duneau et al. (2024) for details). Though the IQP ansatz

would provide non-simulability guarantees for the final model, via the harndness of simulating IQP circuits Bremner et al. (2011, 2016), the ansatz is not universal without post-selections. We hence chose to use a more expressive ansatz, at the cost of possibly learning a classically simulable model.

Any single-qubit boxes are implemented using the Euler parametrization (Figure 4.2f), which is fully expressive using 3 parameters. We implement all states as unitaries applied to a fixed product state ($|00...0\rangle$).

4.1.2 Implementation

We chose to only consider a subset of the possible choices permissible due to time limitations. The options considered are summarised in Table 4.1. A diagrammatic depiction of the model functor is given in Figure 4.3.

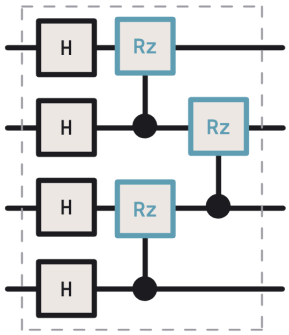
Name	**Ansatz**	**Layers**	**Qubits** (per noun)	**Ancillas** (per channel)	**Channels**
foll	Sim4	3	1	1	$\{follows\}$
babi_{Q}	Sim4	3	1	-	$\emptyset$

Table 4.1: Quantum functor specifications used in this work.

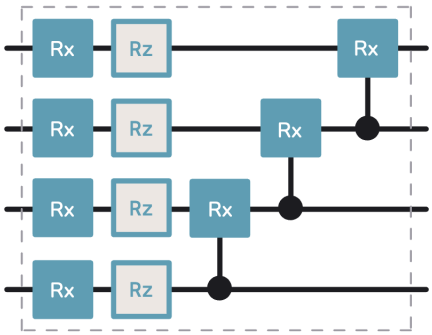
4.1.3 Future work

Bonus question formats Quantum is unique in that more data can be stored in the model than extracted at any given time. Assuming we are to make a quantum implementation, we may then chose to extend the sorts of questions we allow to account for this, for example by building in a 'dense-decoding' type of question, inspired by Farkas et al. (2024).

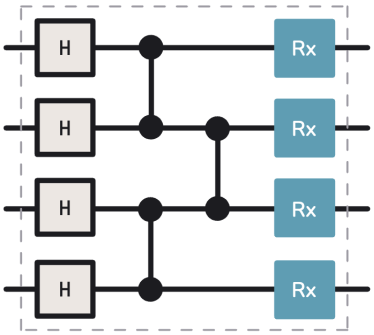
Alternatives Limiting ourselves to quantum circuits in this way is a choice and by no means a requirement. We could instead have chosen to use tensor networks in their full generality for better expressivity, or go the other way and restrict ourselves to easy-to-simulate classes of quantum circuits such as Clifford circuits or free-fermion implementations. Exploration of these cases is left for future work.



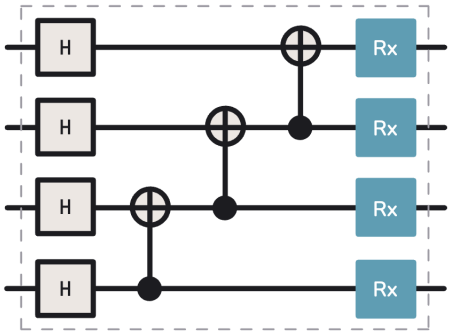
(a) IQP



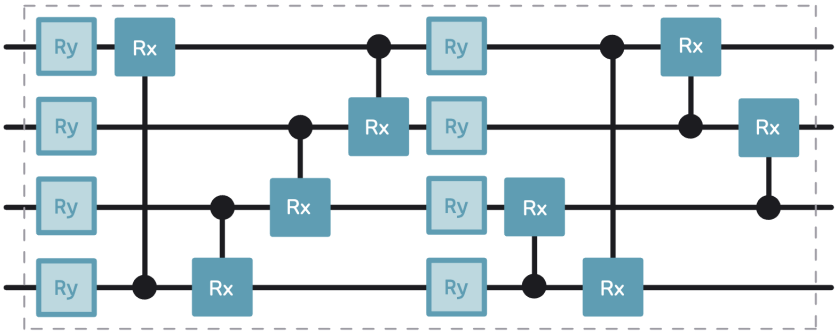
(b) Sim4



(c) Sim9



(d) Sim9Cx



(e) Sim14



(f) Euler Decomposition

Figure 4.2: Quantum circuit ansatze considered in this work. Parametrized gates are highlighted in blue, and the circuit diagrams are read left to right. (a-e) A single layer of the ansatze, for a four-qubit gate. (f) The Euler decomposition used to parametrize single-qubit gates. Notice that the two-qubit gates in IQP and Sim9 commute, whereas they do not for the other ansatze, leading to a lower gate depth.

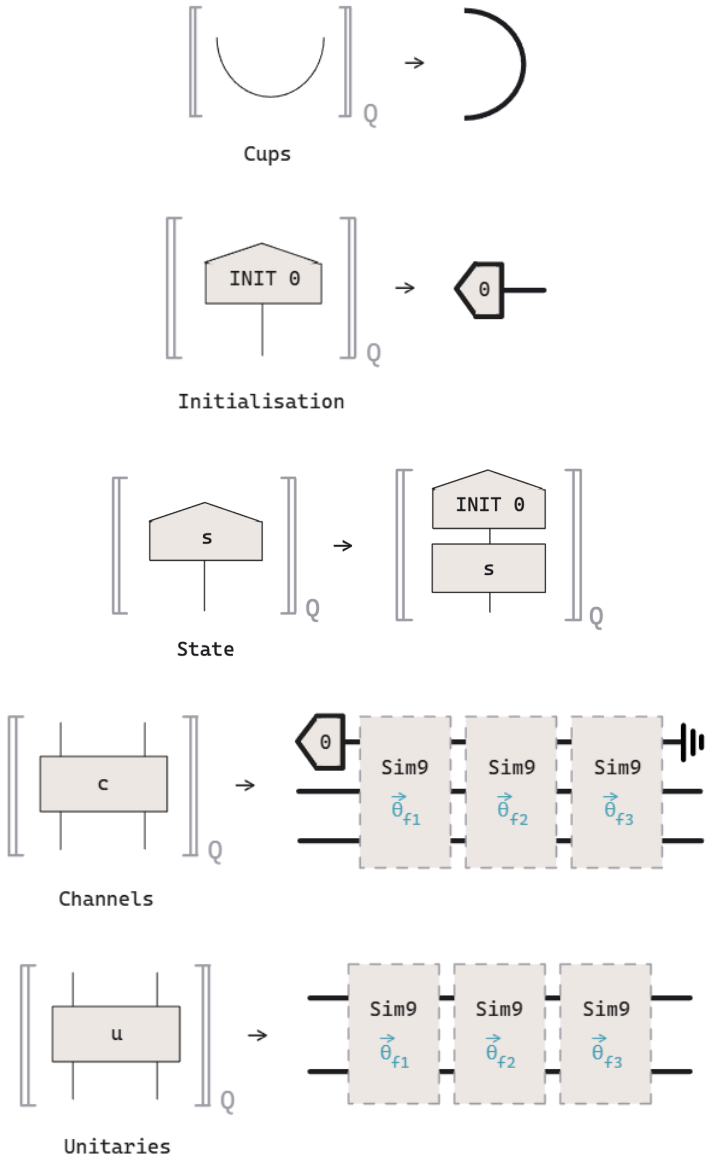


Figure 4.3: Diagrammatic representation of the model functor that applies the quantum ansatz. States are first decomposed as a box applied to a fixed initial state ($|0\rangle$), then the box is expanded as a parametrised Euler decomposition, since the resulting box acts on a single qubit.

4.2 Classical

If we instead chose to use the direct sum ($\oplus$) over the usual tensor product ($\otimes$), we can take our semantic category to be that of neural networks. We specify the category $\mathcal{NN}$ as follows, with a diagrammatic presentation given in Figure 4.4. Starting from $Fd\mathcal{V}ect$:

- Objects are the natural numbers $\mathbb{N}$, including 0.
- Morphisms are matrices $M \in \mathbb{R}^{A,B} : A \to B$.
- The monoidal product $\oplus$ is the cartesian product; equivalently the direct sum. This acts as addition (with unit 0) on the objects.

Comparison

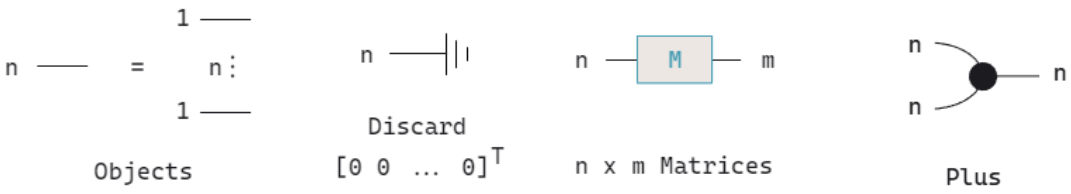
When the monoidal product is the cartesian product, there is only one effect: the terminal object. With no further modifications, this would preclude us completely from using DisCoCirc-style questions, as they require a notion of text comparison that is reliant on the presence of more than a single effect. To accomodate this, we can add a special family of morphisms that act as a pseudo cup, allowing us to append DisCoCirc style questions. It computes the inner product of its inputs. We can specify it recursively in terms of a non-linear function, which ensures compatibility with the monoidal product (as illustrated in Figure 4.5b).

Non-linearities

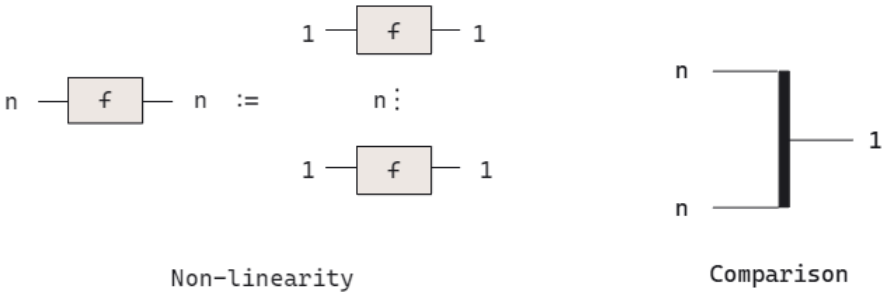
A typical feed-forward neural network is the repeated sequential composition of a matrix layer followed by a non-linearity. We hence add families of *element-wise, non-linear operations*, as these interact nicely with the monoidal product. For an arbitrary function $f : \mathbb{R} \to \mathbb{R}$, generate a family of functions

$$\{F_A : A \to A :: [a_1, ..., a_n] \Rightarrow [f(a_1), ..., f(a_n)] \mid A \in \mathbb{N}\}$$

such that $F_{A \oplus B} = F_A \oplus F_B$. In practice, we will only be interested in certain families, namely ReLu and Mish.

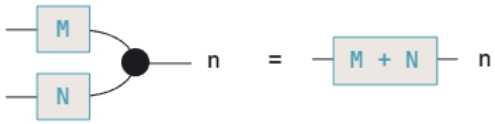


(a)

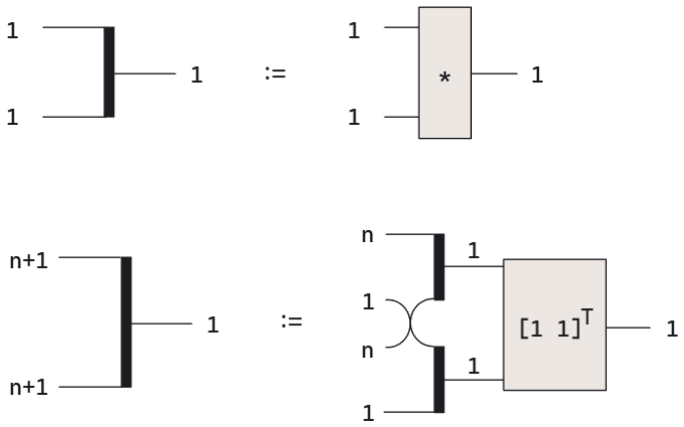


(b)

Figure 4.4: Generators for $\mathcal{NN}$. Diagrams are read left to right. (a) Linear morphisms from $\mathcal{F}d\mathcal{V}ect$. (b) Added, non-linear morphisms.



(a)



(b)

Figure 4.5: Further box definitions. (a) plus acts as element-wise addition. (b) Recursive definition of comp. The star box $* : 1 \oplus 1 \to 1 :: a \oplus b \implies ab$ is a non-linear operation that multiplies its two inputs together.

4.2.1 Implementation

Similarly to the quantum ansatze, we will consider certain schemas for the neural network configurations.

- Linear: A single fully connected NN layer *without* non-linearities.
- Flat: A single fully connected NN layer.
- Hidden$(d_1, ..., d_n)$: A stack of n + 1 fully connected NN layers, with latent dimensions $d_i * noun_dim$.

Note that the expressivity of a neural network strictly increases when moving from Linear to Flat, and increases (for a fixed hidden dimension) from Flat to Hidden(n) to Hidden(n, m). Since Barron (1994), it is known that two hidden layers are sufficient to approximate any function to a desired accuracy. Beyond this threshold increasing the depth can be traded for a (potentially exponential) increase in the width of the hidden dimensions (Eldan and Shamir, 2016). We summarise the options considered in Table 4.2. Though less expressive, we consider Linear models as they provide a close analogue to the quantum scenario, where only the choice of tensor product differs. We visualise the functor diagrammatically in Figure 4.7 and Figure 4.8.

Hyper parameter	**Values Considered**
Non-linearity	ReLu, Mish
Configuration	Flat, Linear, Hidden(1), Hidden(2), Hidden(1, 1), Hidden(2, 2)
Dimension (per noun)	2, 3, 6, 12, 24, 36, 50

Table 4.2: Neural Network functor specifications considered in this work.

4.2.2 Future work

An alternative classical baseline to consider is that of stochastic matrices. These can be expressed as tensor networks, and capture the notion of (merely) classical correlations between systems.

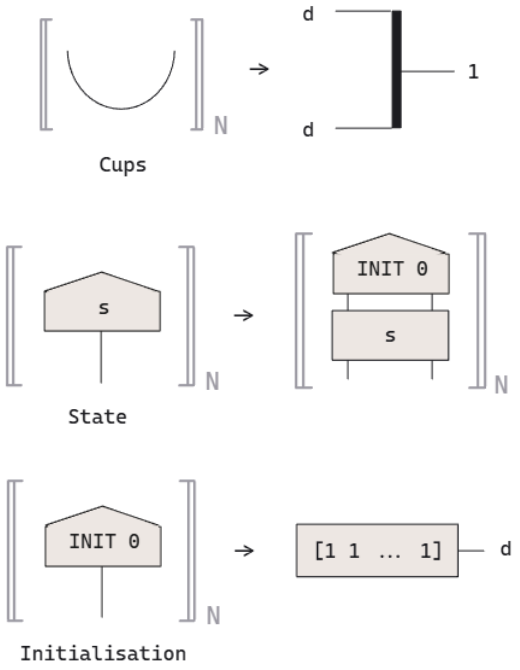


Figure 4.6

Figure 4.7: Diagrammatic representation of the classical model functor. d is the dimension per noun, while h_j is the jth hidden dimension (also per noun).

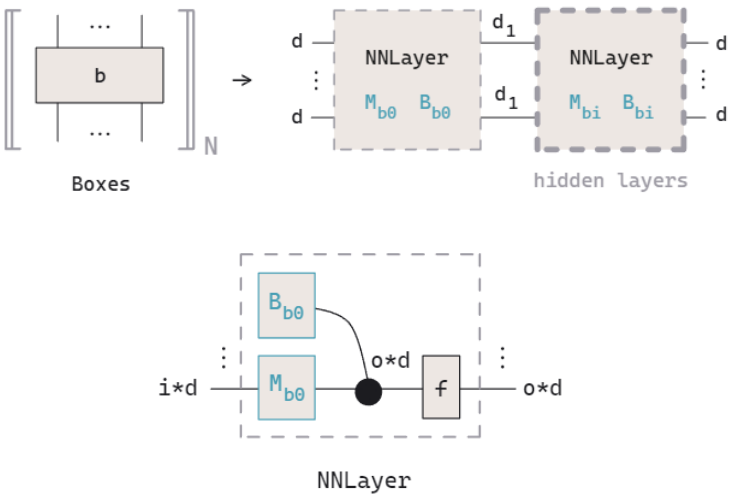


Figure 4.8: The decomposition of a single layer as a simple feed-forward network. f is the chosen activation function. We assume i noun wires go in and o nouns wires come out. The black dot acts as element-wise addition on a pair of matrices, such that each layer calculates $\vec{y} = \mathbf{M}_{b_i}\vec{x} + \mathbf{B}_{b_i}$.



CHAPTER 5

Training methodology

In this section, we present the exact methodology used to prepare the datasets and train the models.

This chapter outlines the theory and experimental methodology used. The code used to generate the results in this thesis is being integrated into the open source NLP package `lambeq`[1]. We additionally provide resources for running equivalent experiments, as well as investigating the trained models explored in this work. The following are made available in the repository at https://github.com/DNA386/Compositional_reading_comprehension_with_DisCoCirc:

Trained models A showcase app allowing interaction with the trained following models is provided.

Example notebooks Tutorials covering dataset generation and model training for both the following and bAbI 6 tasks can be found under the showcase repository, adapted to make use of the `lambeq` package. We also provide notebooks for model analysis, as conducted in Chapter 7.

5.1 Pipeline

The first step required is to convert the text-based inputs into DisCoCirc diagrams. These base diagrams must then be converted according to the choices made by the model - this includes any built in semantic rewrites, frame expansions and the

[1]The package documentation can be found at https://docs.quantinuum.com/lambeq/index.html

choice of ansatze. This is also where the style of question is chosen. The result is then a collection of parametrised diagrams (quantum circuits in the quantum case, or neural networks in the classical case). Figure 5.1 provides a step-by-step view of the processing steps applied.

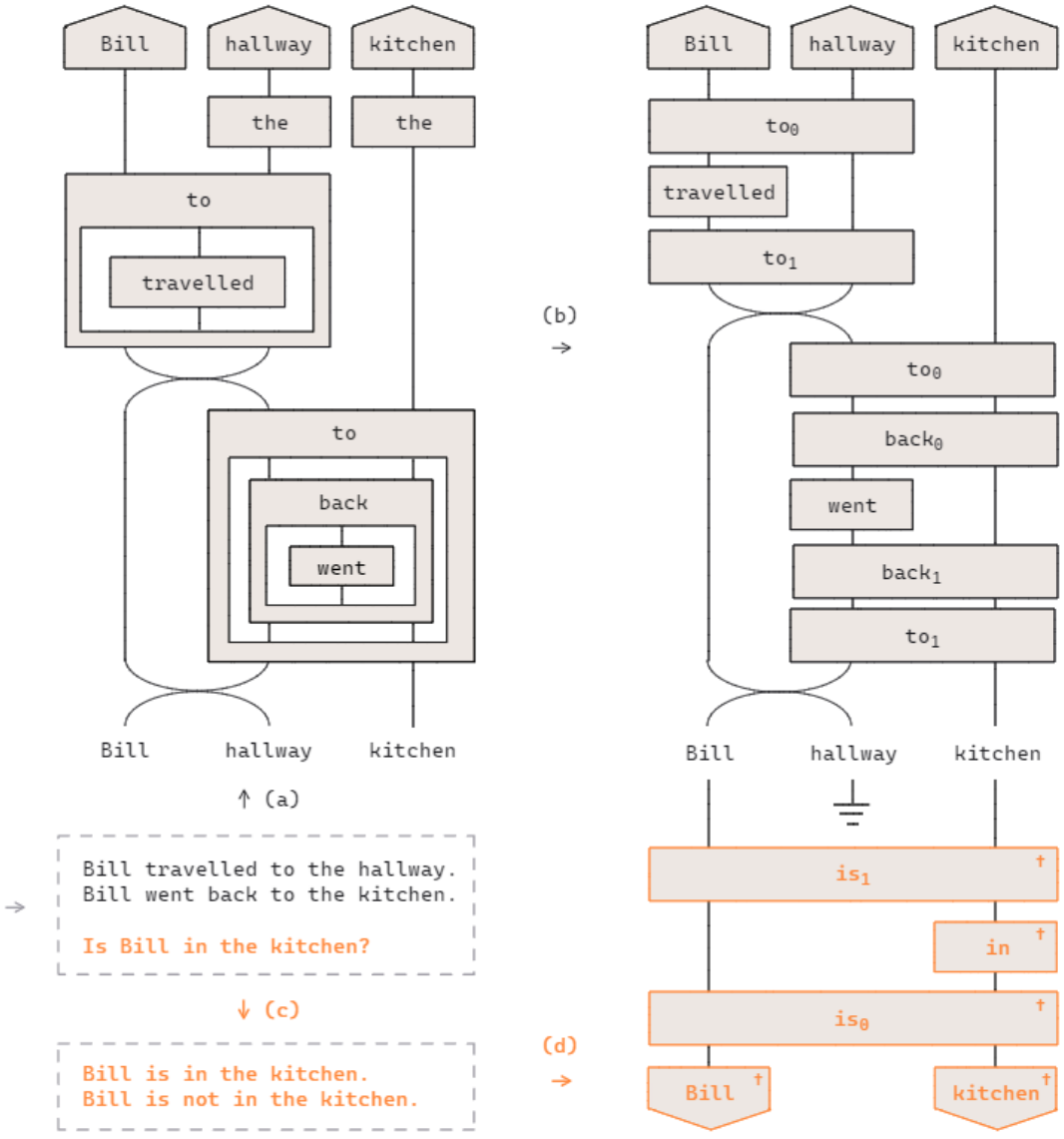


Figure 5.1: Converting an example data entry into a parametrized quantum circuit. Starting from the text dataset entry, we convert the text into a DisCoCirc diagram (a), then apply semantic rewrites and frame expansions (b). Assuming a DisCoCirc question format, we find the assertions that correspond to the question (c), then parse these as with (a), (b). The resulting diagram is then daggered and connected to the appropriate wires in the text diagram, whilst discarding the other wires (d). We then apply the ansatze to the resulting diagram to obtain a quantum circuit (not shown).

5.2 Training

The quantum circuits are converted into tensor networks, which are then evaluated exactly via tensor contraction using `Tensornetwork` (Roberts et al., 2019) on Quantinuum's *duvel4* server. Neural networks are implemented directly as `PyTorch` (Paszke et al., 2019) models with no inputs. We use `PyTorch` to track the parameters and the Adam optimizer (Kingma and Ba, 2015) to train them. The hyper-parameters are tuned using `Ax` (Bakshy et al., 2018; Facebook) where applicable.

For each training epoch, the model is evaluated on the entire training dataset in batches, with a parameter update occurring once per batch. We save the parameters obtained at the end of each epoch and record the loss and validation accuracy. Every three epochs, starting from the first, we evaluate and record the accuracy of the model on the training dataset[2]. For each training run, we select the model from the epoch with the best score (as defined by the choice of training regime below), tie-breaking if necessary with the closest logged training accuracy, then loss.

5.3 Optimising compositionality

Here we present the format of the experiments conducted. We first give an overview of the various training regimes considered, then provide more details of the concrete implementations for each dataset.

5.3.1 Training regimes

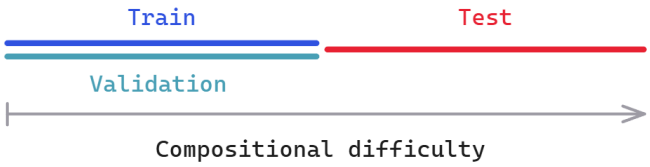
We consider a selection of training regimes, in the interest of finding one that has a better tendency for producing models that compositionally generalise.

Vanilla

As a baseline, we consider the *vanilla* approach: the data is split into three segments called *Train*, *Validation* and *Test*. The model is trained on the training instances, with the validation set being evaluated at each epoch. The final model selected is then the one with the highest validation accuracy. We enforce that the *Validation* set is generated from the same distribution as the *Train* set, whilst

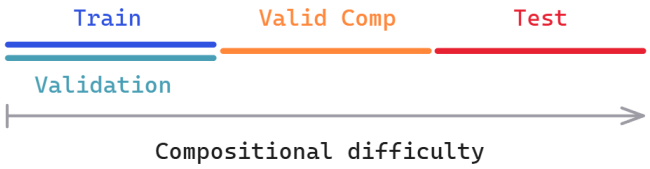
[2]We do not always do this every epoch in order to keep model training times down.

the *Test* set will in general contain items from a broader distribution to test compositional generalisation. Note that in many other ML setups, the test set is *also* drawn from an identical distribution to that of the *Train* (and *Validation*) set(s).



Compositional validation

A straightforward way to aid model selection is to add in a secondary validation dataset, drawn from a tractable subset of the *Test* distribution and known as *Valid Comp*. A small number of promising candidate models can be selected using the vanilla method, and each evaluated on the *Valid Comp* dataset as part of a second round of selections. The model with the best *Valid Comp* accuracy is then selected to evaluate on the test set.

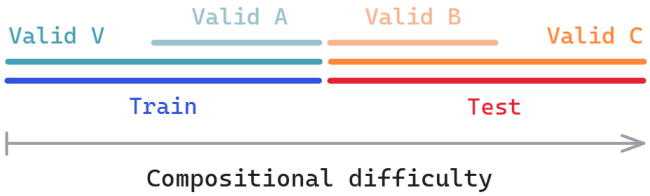


Extended validation

Given the nature of compositionality, a natural extension of the vanilla approach is to introduce some datapoints from the *Test* distribution to the validation set, so that the selected model is one that although still only trained on the *Train* distribution, is one that shows more promise for generalisation. We consider a selection of new validation set candidates, expressed according to the difficulty strata of the datapoints they contain:

Valid V	The vanilla validation set, where the distribution overlaps with the *Train* set.
Valid A	A subset of the *Valid V* set, generally towards the 'harder' end of the distribution.
Valid C	An analogue to *Valid V*, but where the distribution overlaps with the *Test* distribution.
Valid B	A (generally tractable) subset of *Valid C*.
Valid AB	A combination of *Valid A* and *Valid B*.
Valid All	A combination of *Valid V* and *Valid C*.

Note that *Valid AB* and *Valid All* can be used to estimate the compositionality score of the dataset.



Curriculum learning

Rather than changing the contents of the *Validation* set, we might instead alter the *Train set*. In this case, the model is trained on progressively more data, where we gradually include more items requiring compositional generalisation.

Future work: meta-learning

Taking the previous idea further, we can implement a compositional form of metalearning. In this case, we split the data into training and test distributions according to the amount of compositional generalisation they require, then train the model to have high generalisation accuracy from train to test. Fixing the number of samples seen as part of the training step to a low number helps two things: first, the final trained model is as close as possible to the optimized model. Secondly, the costs are minimised (the higher order nature of the training requires gradients to be computed over successive iterations which can blow up the cost of evaluating datapoints; we get the same kind of problem with batching).

Evaluation

We evaluate these approaches according to two main metrics: *Test* accuracy and overall compositionality score. Additionally, we consider the trade-offs between these metrics and the amount of resources required to implement each one.

5.3.2 Following

For the *following* datasets, we consider only productivity in the compositional validation training regime, with additional hyper-parameter tuning. The hyperparameters we considered and their final tuned values are summarised in Table 5.1 below. The tuning was conducted using the four-directional dataset. We did not do any further tuning on the two-directional dataset, as the model already achieved a high accuracy with the first hyperparameters we chose. The initial parameters for the two-directional model were recorded, but not explicitly seeded. The optimisation of the tuning was for the best *Validation* accuracy, and the tuning was carried out using `Ax` (Bakshy et al., 2018; Facebook). Each tuning run was run for 15 epochs. From these runs we selected the best hyperparameters and ran some further training runs for 200 epochs. To select our final model, we first picked 2 candidate models from the tuning runs with high *Validation* accuracy, evaluated both on the *Valid Comp* dataset, and selected the model with the best generalisation performance.

Hyperparameter	**Type**	**Options**	**Final value**	
			Four-directions	Two-directions
Learning rate	range	0.0001 - 0.1	0.02840955	0.005
Batch size	choice	2^0, 2^1, ..., 2^8	256	1
Parameter seed	range	0 - (2^{32}-1)	1151618203	N/A

Table 5.1: Training hyperparameters used and the values we considered when tuning on the four-directional dataset.

We trained two model families, one on each dataset and evaluate their productive generalisation. For the two directional model, we additionally perform 5-fold cross-validation to contextualise the robustness of the results, and highlight the necessity of going beyond vanilla model selection when selecting compositional models.

5.3.3 bAbI 6

For this dataset family, we evaluate the four aspects of compositionality individually to test the model's compositional generalisation.

Compositionality

We train a model each for the quantum, Linear and non-linear neural models, using the extended validation regime with the *Valid AB* scheme, for the productivity, systematicity and substitutivity datasets. Additionally, we train the quantum model using the curriculum regime for each of these datasets, as detailed in Section 5.3.3.

We performed some initial manual tuning on the quantum models, settling on a learning rate of 0.0005 and a single batch for the productivity dataset, which we then transferred to the other datasets without requiring further tuning. Training runs lasted between 20-75 epochs for the quantum models, and from 50 to 200 epochs for the neural models. We document the final parameters for the quantum models in Table 5.2. The quantum models were run for fewer epochs largely due to resource constraints - most of the neural architectures were much faster to run, making more epochs feasible in the time allocated.

Hyperparameter	**Productivity**	**Substitutivity**	**Systematicity**
Learning rate ($\times 10^{-3}$)	5.00	5.00	5.00
Batch size	1	1	1
Max epochs	75	20	30

Table 5.2: Final hyperparameters selected for the neural models. Full tuning was not possible in all cases due to time and resource constraints; for the 'inherited' configurations, hyperparameters from another task were taken, and adjusted with a few rounds of manual tuning if necessary.

For the neural models, we perform extensive hyperparameter tuning, as summarised in Table 5.3, according to the options presented in Table 4.2. The final values selected are summarised in Table 5.4. In cases where no hyperparameter tuning was used, we run the training with 5 different initialisations, keeping the hyperparameters constant, and select the best model out of these 5 runs, otherwise we selected the best model obtained from the tuning runs. For some of the initial tuning runs, the max epochs was set to 200, however we noticed that selected

models were rarely from the last 100 epochs, so capped this further for the final training runs.

Hyperparameter	**Type**	**Options**
Learning rate	range	0.00005, 0.005
Batch size	choice	1, 4, 8, 16, 32, 64, 128
Wire dimension	choice	2, 3, 6, 12, 24, 36, 50
Hidden layers	choice	Linear, Flat, Hidden(1), Hidden(2) Hidden(1, 1), Hidden(2, 2)
Activation	choice	ReLu, Mish

Table 5.3: Values considered for hyperparameters when tuning neural models.

Hyperparameter	**Productivity**		**Substitutivity**		**Systematicity**	
Tuning Method	Ax	Inherit	Ax	Inherit	Inherit	Ax
Learning rate ($\times 10^{-3}$)	4.97, 5.00	5.00	3.60	5.00	5.00	50.0
Batch size	16, 4	4	1	4	4	1
Max epochs	50	50	100	50	50	100
Wire dimension	36	12	24	12	36	2
Hidden layers	Linear	Flat	Linear	Flat	Linear	Hidden(1)
Activation	-	Mish	-	Mish	-	Mish

Table 5.4: Final hyperparameters selected for the neural models. Full tuning was not possible in all cases due to time and resource constraints; for the 'inherited' configurations, hyperparameters from another task were taken, and adjusted with a few rounds of manual tuning if necessary.

Overgeneralisation We test overgeneralisation in a similar way as the other tasks, however focus on only a single quantum and neural model architecture. For the neural model, we use the same hyperparameters as the productivity task, selecting the Flat architecture. Likewise, we pick the same quantum architecture as for proudctivity, but capped the max epochs at 30.

Cross-validation We conduct a more in-depth analysis of the results on the productivity dataset. For each architecture choice investigated we run 5-fold cross validation to investigate the variability of the results. A different number of max epochs than for tuned models was used to keep the evaluation time more feasible.

We capped the quantum models at around 2.5 hours per run, leading to 40 epochs for the basic quantum model, and 60 for the curriculum models. For the neural architectures, we allowed 100 epochs, as each run took under 30 minutes.

Extended validation

We investigate the effect of different extended validation schemes for the cross-validated quantum productivity models. To evaluate this, we consider the training runs from the cross-validation. For each run, we compute the validation accuracy at each epoch for a series of validation sets. These sets are selected such that they never overlap with any training data. We then select the models with the best respective validation accuracies, and compare their performance on the test set.

Curriculum

Finally, for the quantum architecture, we conduct an initial exploration of curriculum learning. This introduces some new hyperparameters, concerning how new elements are selected and introduced to the training data. We used the training strata as curriculum levels, such that the model is exposed to progressively more difficult datapoints during training. We controlled this in two ways: first, by varying the number of epochs over which the data is to be introduced, and second, whether the elements in the next stratum were introduced gradually, or all at once after a number of epochs. The hyperparameters were tuned manually, with the final values summarised in Table 5.5.

Hyperparameter	**Considered**	**Value**
Learning rate	0.0002, 0.0004, 0,0005	0.0005
Batch size	1, 4, 8	1
Curriculum epochs	6, 10, 15, 25, 30, 40	30
Max epochs	30, 50, 75	75
Curriculum step	smooth, step	smooth

Table 5.5: Hyperparameters considered and final values used for curriculum learning.

Part III

Results



CHAPTER 6

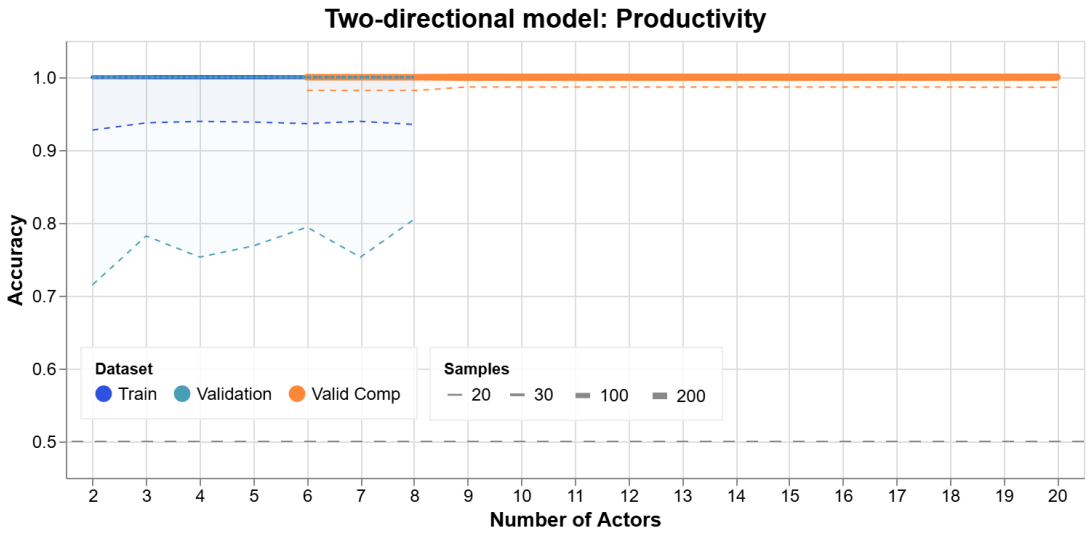
Compositional generalisation

Having trained our compositional models, we may now test whether they manage to compositionally generalize. We consider and compare the model results over the two datasets (*following* and *bAbI 6*), including a comparison of the quantum and classical models. Additionally, we conduct a small analysis of different training regimes for the purpose of maximising compositional generalisation, concluding that an extended validation set is required for more reliable training.

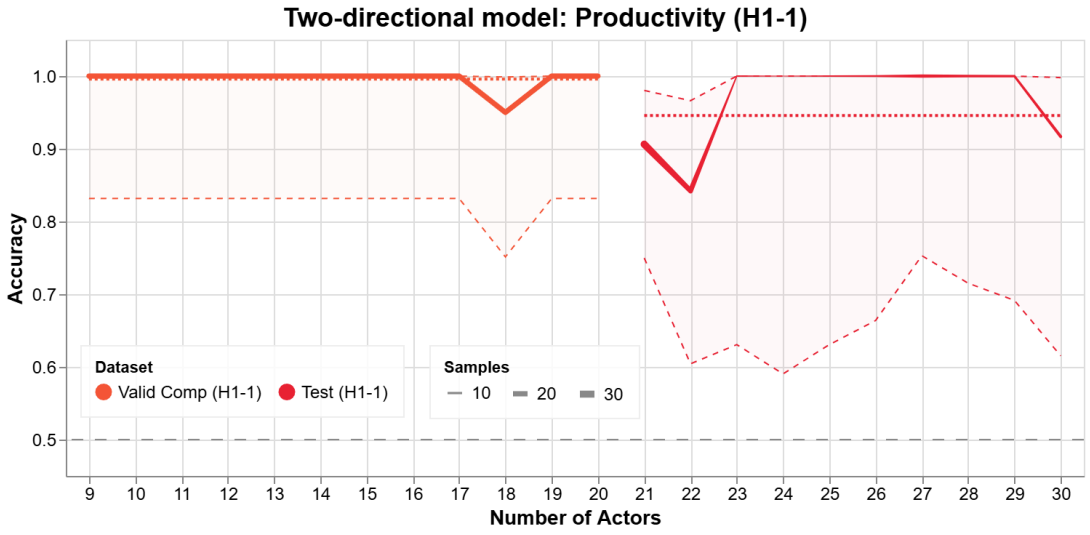
Having trained models for each of the tasks considered, we may now consider whether they have appropriately learnt to be *semantically* compositional, and thus generalise to examples outside the training distribution.

6.1 Proof-of-concept: following

For the *following* dataset, we focus on productivity. Recall that the texts present in the *Train* set include up to 8 nouns. Testing productivity entails observing whether the *Test* accuracy is correlated with the size of the stories evaluated. During the in-task training of the DisCoCirc model, we used only exact tensor contractions. We also used exact tensor contractions to obtain the *Valid Comp* accuracy for instances featuring up to 20 nouns. Pushing beyond 20 nouns, we use Quantinuum's H1-1 to evaluate accuracy on *Test (H1-1)* for instances up to 30 nouns from the two-directional dataset, and used shot-based noisy simulation (as described in Duneau et al. (2024), App. I) to obtain the results for the four-directional *Test (qujax)* dataset.

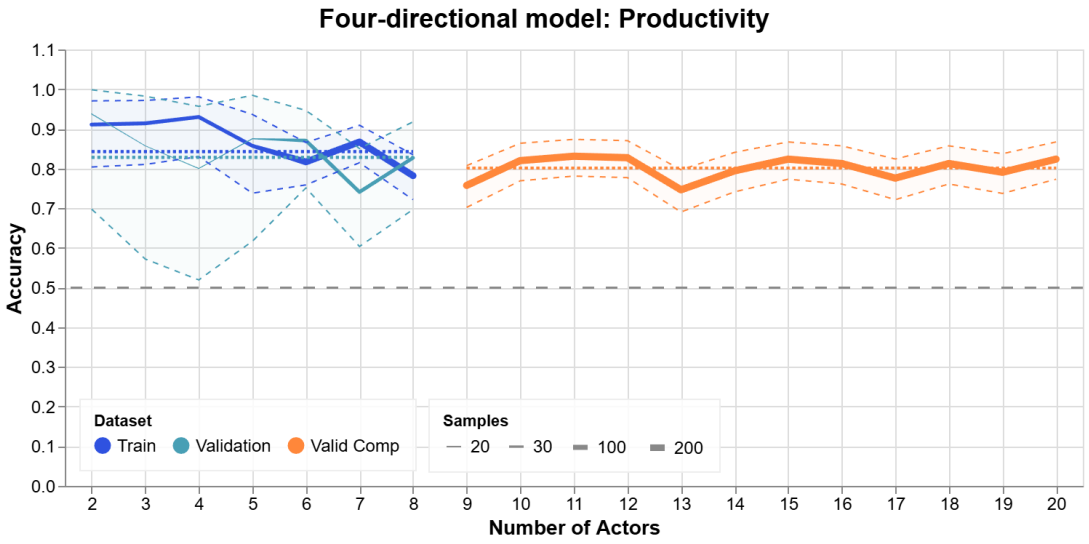


(a)

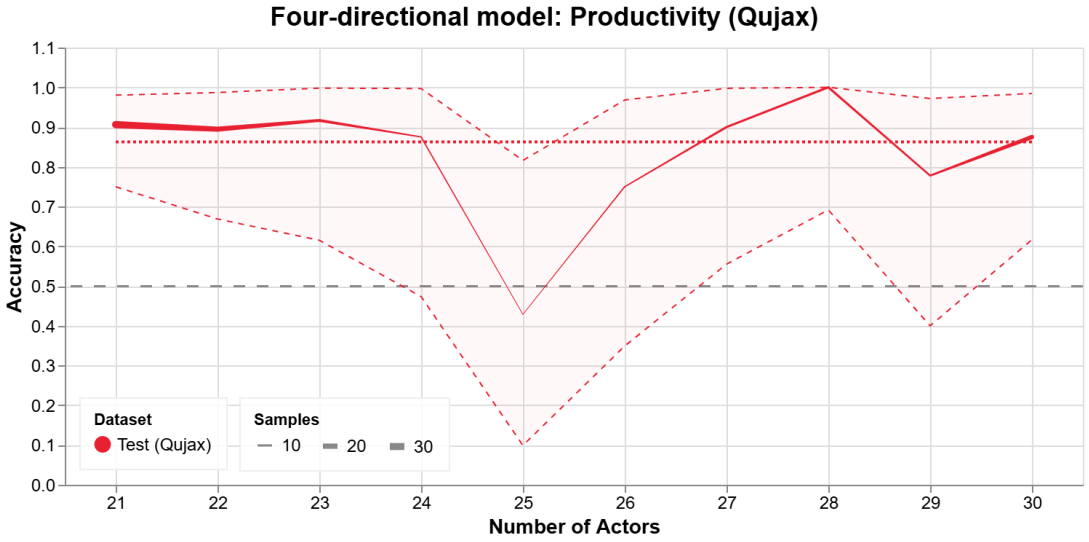


(b)

Figure 6.1: Visualising the two directional model's capacity for compositional generalisation, per number of actors for (a) exactly simulated datapoints and (b) datapoints evaluated on Quantinuum's ion-trap quantum computer H1-1. The error bars capture the 95% Clopper-Pearson confidence interval (Clopper and Pearson, 1934) for the mean, taking into account the number of samples available. Results are plotted as lines with shaded error regions. *Train, Validation*, and *Valid Comp* datapoints were simulated, going up to 20 actors, while *Valid Comp (H1-1)* and *Test (H1-1)* datapoints were evaluated on H1-1 using 50 shots per circuit. The number of datapoints sampled for each number of nouns is reflected by the thickness of the line. For the *Valid Comp (H1-1)* and *Test (H1-1)* datapoints, there were only a few samples available for each number of actors, leading to wider error margins. The dashed grey line represents the baseline obtained by random guessing. The average performance per dataset is shown as a dotted line for each split.



(a)



(b)

Figure 6.2: Visualising the compositional generalisation of the four directional model, per number of actors for (a) exactly simulated datapoints and (b) datapoints evaluated via shot-based simulation with `qujax`. The error bars capture the 95% Clopper-Pearson confidence interval (Clopper and Pearson, 1934) for the mean, taking into account the number of samples available. Results are plotted as lines with shaded error regions. *Train*, *Validation*, and *Valid Comp* datapoints were simulated exactly, going up to 20 actors. The *Test (Qujax)* performance of the model was simulated with noise using `qujax` with 1000 shots per circuit. The number of datapoints sampled for each number of nouns is reflected by the thickness of the line. For the *Test (Qujax)* datapoints, there were only a few samples available for each number of actors, leading to wider error margins. The dashed grey line represents the baseline obtained by random guessing. The average performance per dataset is shown as a dotted line for each split.

With statistical significance, the accuracy of the model on the *Valid Comp* set does not decay for the two- and four-directional datasets in Figure 6.1a and Figure 6.2a. This also happens for the two-directional *Test (H1-1)* sub-dataset in Figure 6.1b. For the four-directional *Test (qujax)* sub-dataset in Figure 6.2b, we note that the low number of sampled data points for each text width led to larger confidence intervals. Additionally, for the drop at 25 actors, approximately half of the samples selected were amongst those we expect the model to solve incorrectly a priori (see Chapter 7 for a discussion of this point), which is a much higher proportion than in the rest of the dataset.

We argue that this is experimental evidence that compositionality enables generalisation, as to perform inference we need only compose pre-trained components. In other words, we have provided experimental evidence that the quantum word circuits have learned effective semantically compositional representations.

6.1.1 Cross-validation

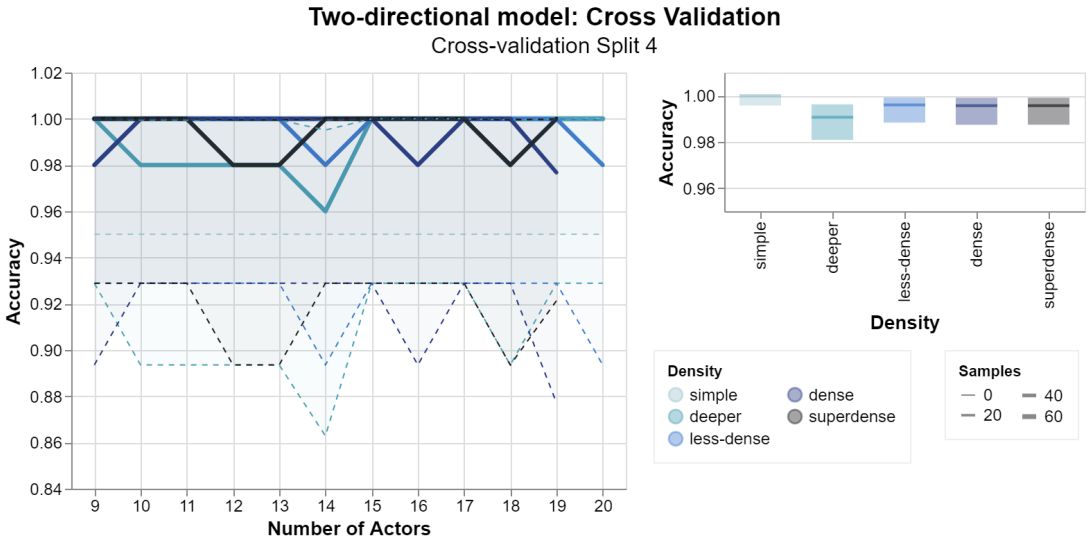


Figure 6.3: Cross-validated compositional generalisation for the two directions dataset on the *Valid Comp* dataset, for cross-validation split 4 of 5. The shaded regions show the 95% Clopper-Pearson confidence interval for the average accuracy of each density and noun width (number of actors), with edges marked by dashed lines.

We perform 5-fold cross-validation using the hyperparameters chosen for the two-directional dataset, with 5 iterations per fold. Each model was trained for 10 epochs. For each fold, we pick the model with highest *Validation* accuracy and evaluate it on the *Valid Comp* dataset. All but one model managed to generalise

perfectly, matching the performance of our final model. The results for the model that did not generalise perfectly are summarised in Figure 6.3. From this we see that the generalisation capacity of the model is broadly independent of the initialisation, however further selection beyond *Validation* is required to ensure that the final model generalises *compositionally*: in our case, although all models achieved 100% accuracy on the *Train* and *Validation* datasets, we see that the last model failed to generalise as well on the *Valid Comp* dataset.

6.1.2 By density

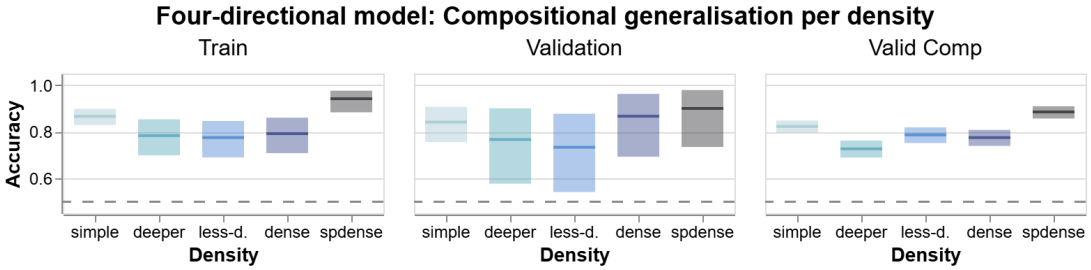


Figure 6.4: Compositional generalisation of the four-directional model per density and dataset. The model was trained on all datapoints up to 8 nouns. The shaded regions represent the 95% confidence intervals for the mean, with edges highlighted by dashed lines. We also plot the random guessing baseline at 50% with a dashed grey line. See Figure 6.8 for a breakdown by number of actors.

Here we give a more detailed breakdown of the compositional generalisation curves, looking at each density separately. Figure 6.4 displays the four-directional model. The two-directional model is not shown, as it achieved full accuracy. Additionally, we show the productivity score, computed according to Equation CS using the *Train* and *Valid Comp* datasets in Figure 6.5.

Recall from Section 3.3.1 that the density is approximately correlated with the simulability of the dataset, with more dense sets being harder to simulate. This does not necessarily correlate with the difficulty of answering questions about such stories, however. In Chapter 7, we will study the model in more detail and account for the trends exhibited here.

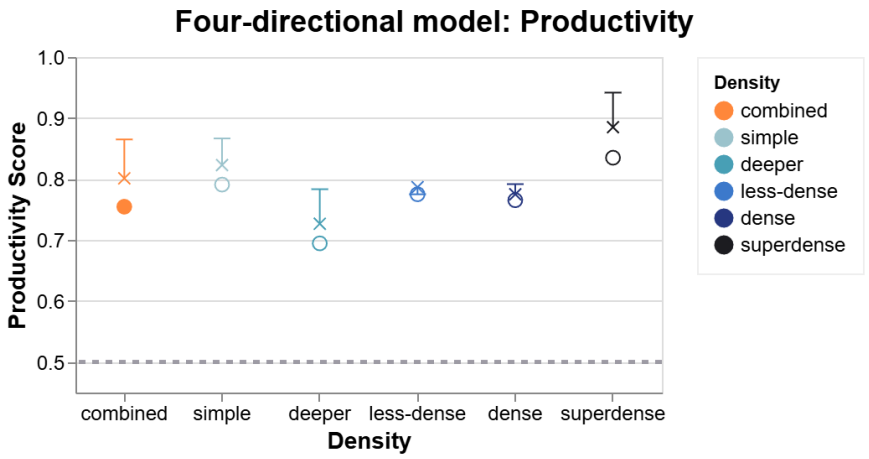
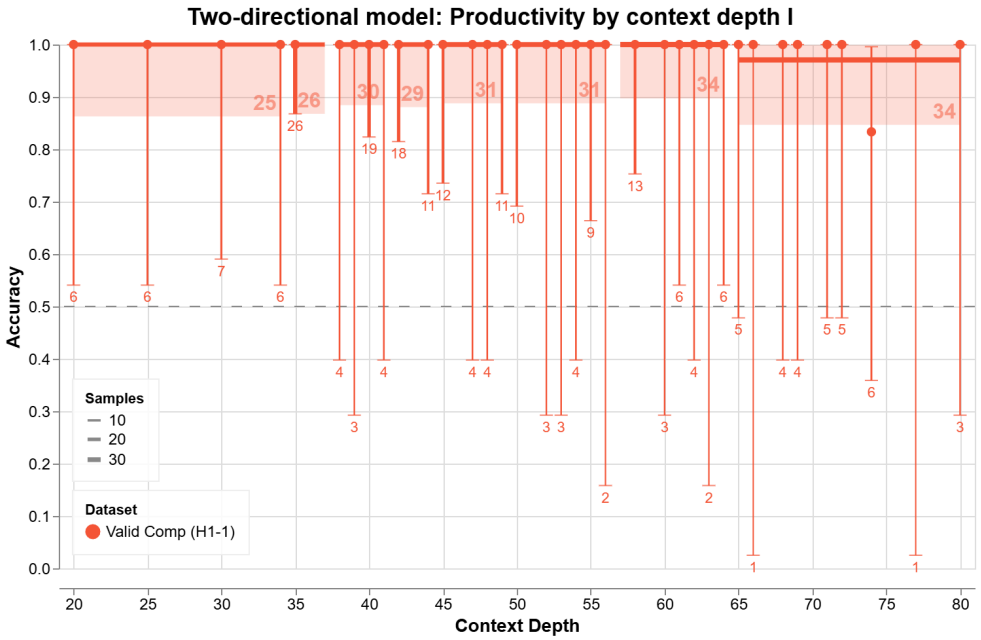


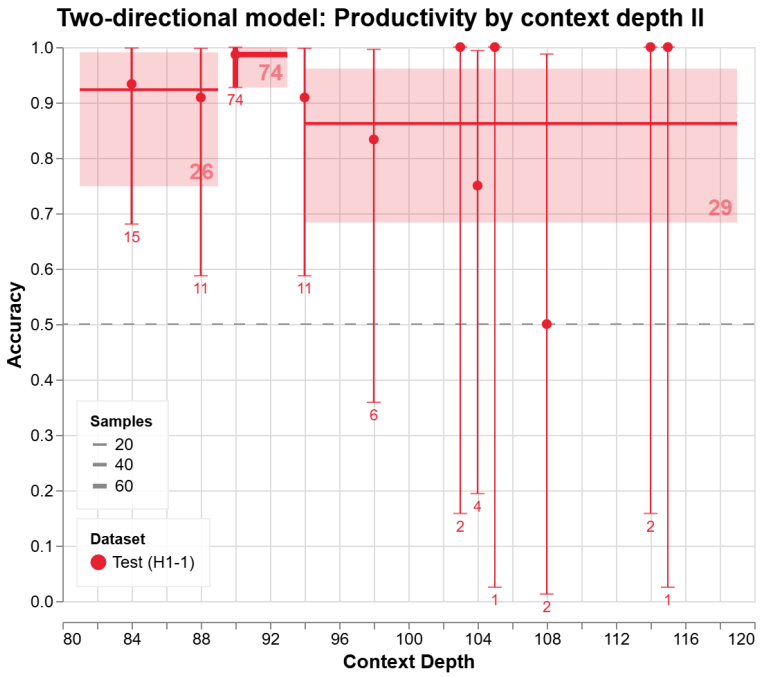
Figure 6.5: Quantifying the compositional generalisation of the four-directional model via productivity score. The score is displayed for each density as well as the combined score over all datasets. The productivity score is displayed as a point, while the *Train* and *Valid Comp* accuracies used to compute the score are plotted as a dash and cross respectively, and linked by a line. The dashed grey line represents the random guessing baseline.

6.1.3 H1-1 and *Test (Qujax)* sub-datasets

Both the four- and two-directional datasets correlate the number of actors with text depth, however both the *Test (H1-1)* and *Test (Qujax)* subsets that were actually evaluated were sampled from those that could be compiled down to 20 qubits from the dense and superdense datasets. Figure 3.4a visualises the subset of datapoints that were sent to the quantum computer H1-1, and Figure 3.4b visualises the samples from the four-directional dataset. We see that above 20 actors, there is instead an anti-correlation between number of actors and text depth, though all instances had text depths above that seen in any of the *Valid Comp* instances. We visualise compositional generalisation in terms of the text depth directly in Figure 6.6 and Figure 6.7 for the samples sent to H1-1, and the four-directional *Test (Qujax)* set. Of particular note is that we had few samples for each depth, so the confidence intervals for the mean are very wide, however in the cases where more samples were present the mean remains high and above random guessing.



(a)



(b)

Figure 6.6: Compositional generalisation for the H1-1 samples per sentences in the text for (a) the *Valid Comp (H1-1)* and (b) *Test (H1-1)* datasets. The error bars shown represent the 95% confidence interval for each depth, calculated using the exact Clopper-Pearson method, which is sensitive to the number of samples. Since the number of samples per depth is very uneven, we also plot confidence intervals for binned data, targeting around 30 samples per bin. The average per bin is drawn as a line covering the binned region, with a shaded confidence interval. The number of datapoints per bin is noted inside each binned region.

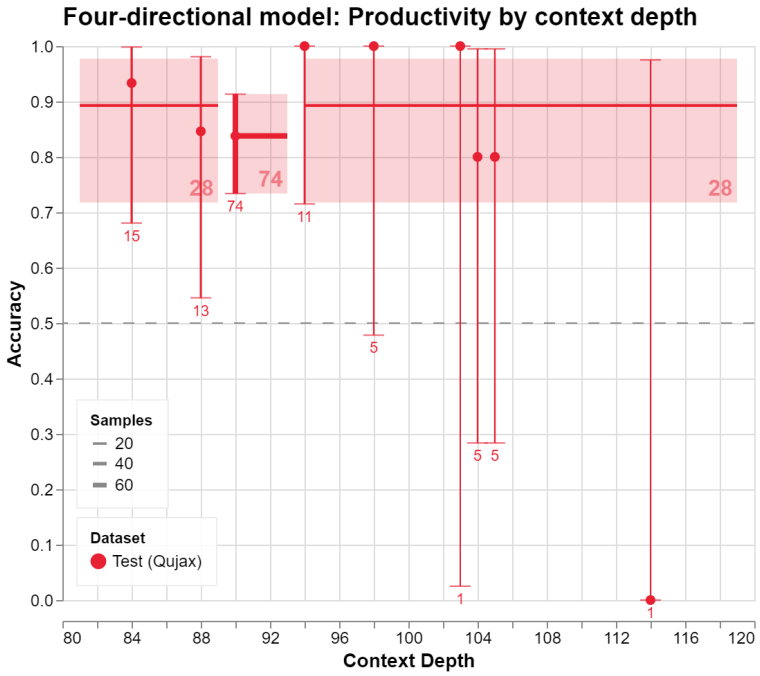


Figure 6.7: Compositional generalisation for the four-directional *Test (qujax)* samples, per number of sentences in the story. The error bars shown represent the 95% confidence interval for each depth, calculated using the exact Clopper-Pearson method. As with Figure 6.6, we also plot the mean and confidence intervals for binned data, using the same bins as for the *Test (H1-1)* dataset. The average per bin is drawn as a line covering the binned region, with a shaded confidence interval. The number of datapoints per bin is noted inside each binned region.

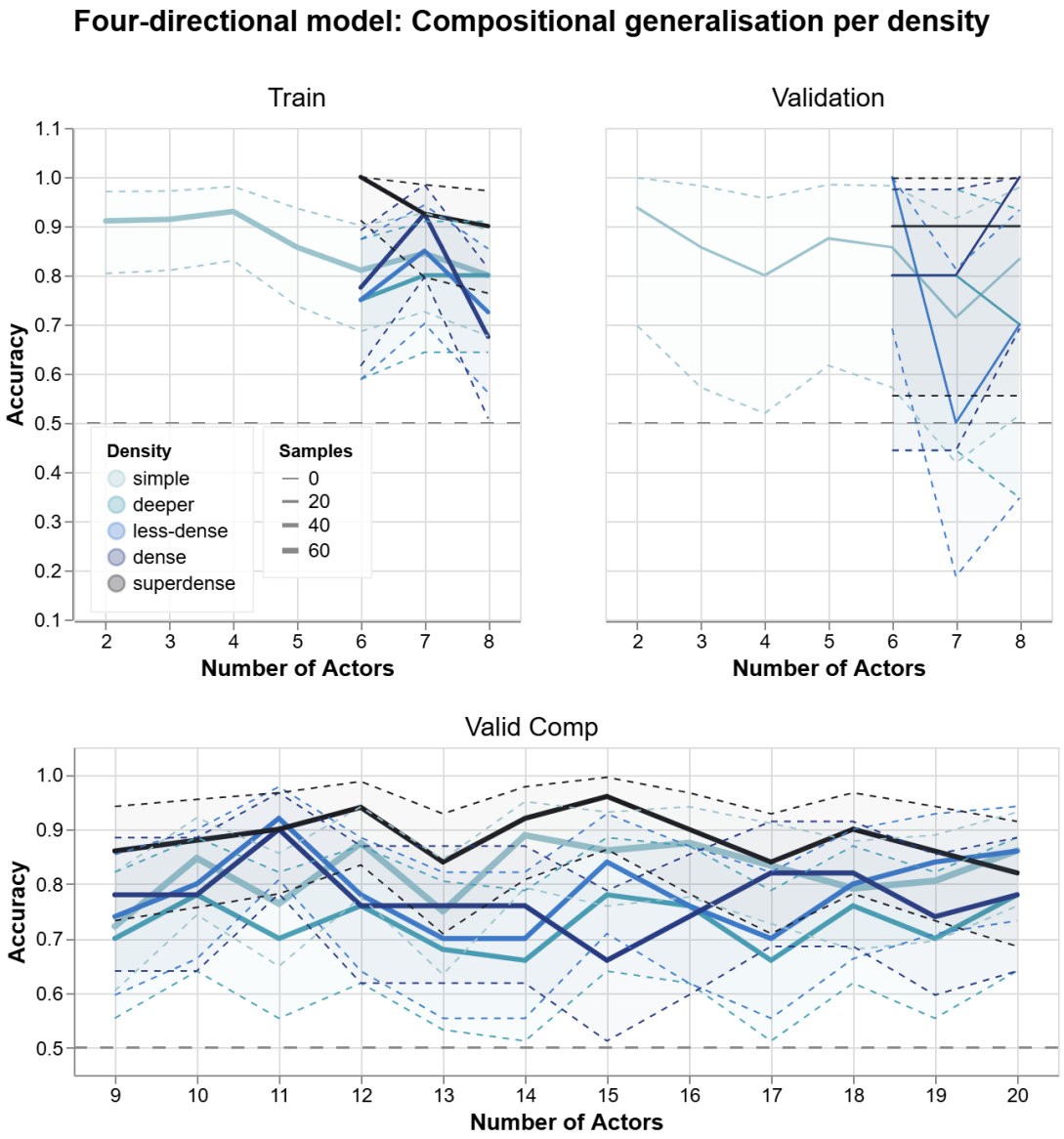


Figure 6.8: Compositional generalisation of the four-directional datasets per density, visualised per context depth, for the *Train*, *Validation* and *Valid Comp* datasets. The model was trained on all datapoints up to 8 nouns. The shaded regions represent the 95% confidence intervals for the mean, with edges highlighted by dashed lines. We also plot the random guessing baseline at 50% with a dashed grey line. Refer back to Figure 6.4.

6.2 Aspects of compositionality: bAbI 6

Moving to the bAbI 6 dataset, we now consider systematicity, substitutivity and overgeneralisation in addition to productivity. We also investigate which training methods help to maximise the chances of obtaining (and selecting) a compositional model, for productivity. Figure 6.9 summarises the compositionality scores (as defined in Equation CS) achieved by the models for the productivity, substitutivity and systematicity. Recall that localism is not explicitly measured, as the model is local by construction for the fragment of English used in the bAbI 6 dataset. Overgeneralisation is studied separately in Section 6.2.1.

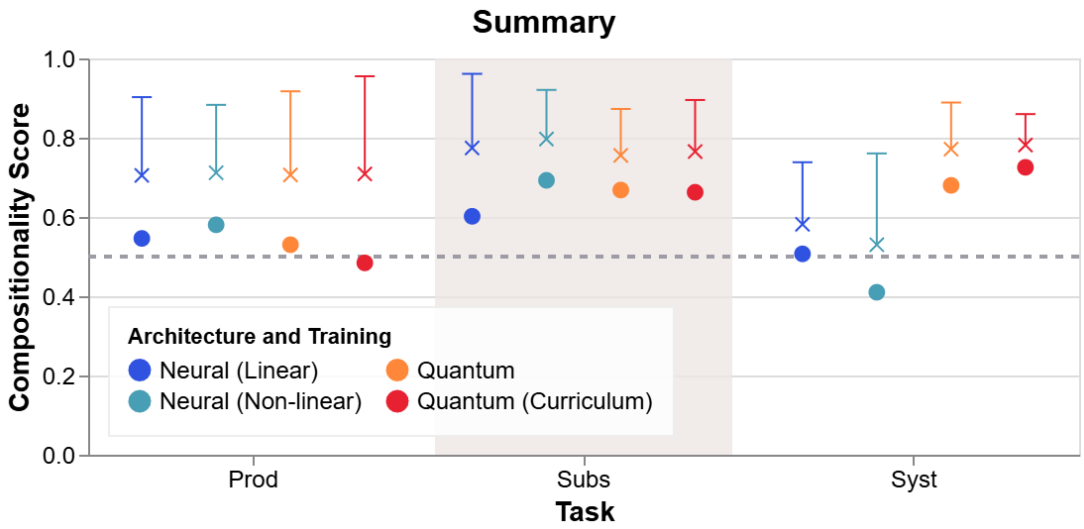
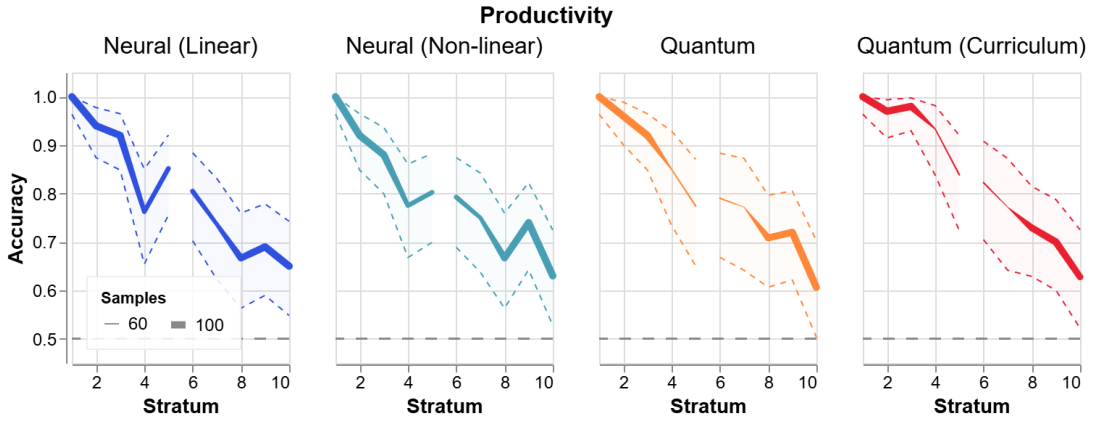
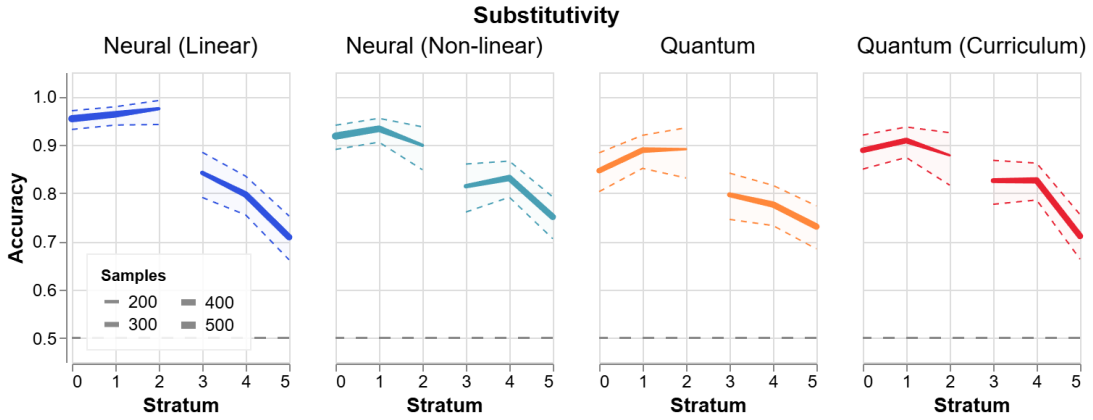


Figure 6.9: Summary compositionality scores achieved by the best model selected for each task by the quantum and neural DisCoCirc models considered. For each dataset, we plot the model's compositionality score as a point. The *Train* and *Test* accuracies used to compute the productivity score are plotted as a dash and cross respectively, and linked by a line. Recall that the compositionality score is a function of the difference between the training and test accuracies, multiplying the training accuracy.

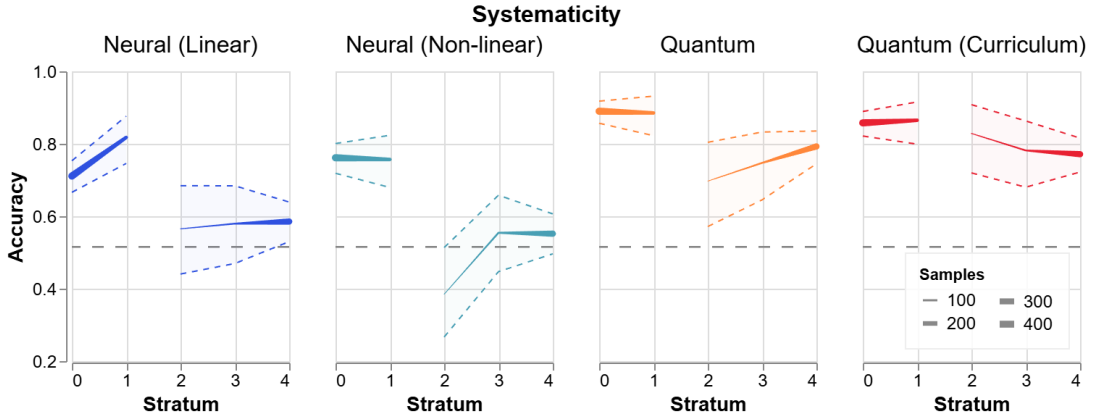
Productivity For the productivity task, all of the models scored very similarly on the *Test* set, such that the relative difference between the training accuracies was the main source of variation in the final productivity score. We plot the scores per productivity stratum (context depth) in Figure 6.10a.



(a)



(b)



(c)

Figure 6.10: Visualising the accuracy per stratum for the models trained on the (a) productivity, (b) substitutivity, and (c) systematicity datasets. The shaded regions represent the 95% Clopper-Pearson interval for the mean. The thickness of the line reflects the number of datapoints available at each stratum. We display only the scores for the *Train* and *Test* datasets.

Substitutivity As visualised in Figure 6.10b, the models all again perform similarly, though the Linear model seemed to overfit the training set more, leading to a worse compositionality score. Compared to the productivity task, the models exhibit a lower drop in accuracy from the *Train* to the *Test* set, however we still find that the accuracy seems to decay as the number of synonyms encountered increases.

As visualised in Figure 6.11, we see that this may be linked to the models' struggles with productivity, especially for the quantum models. (Recall from Figure 3.14a that the substitutivity strata are somewhat correlated with the context depth.)

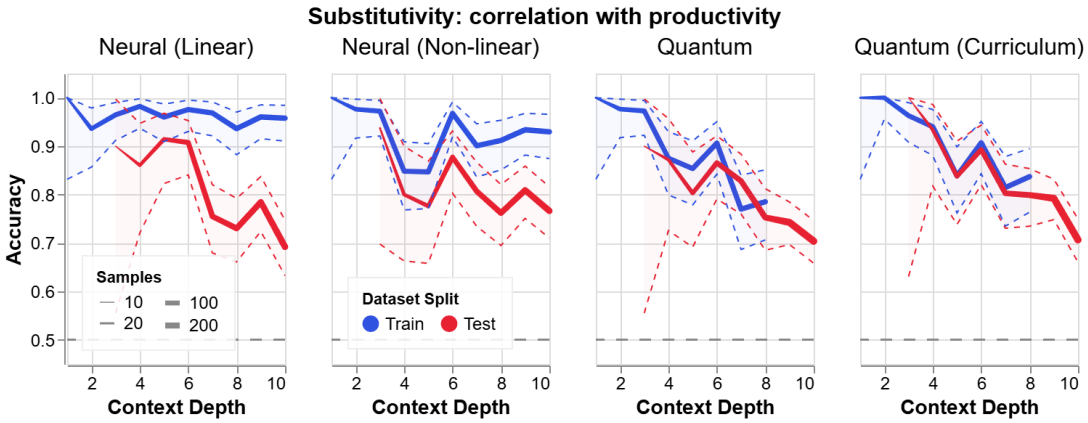


Figure 6.11: Visualising the substitutivity models' accuracy per productivity stratum (context depth). The shaded regions represent the 95% Clopper-Pearson interval for the mean. The thickness of the line reflects the number of datapoints available at each stratum. We display only the scores for the *Train* and *Test* datasets.

Systematicity This task showed the most variation between the architectures: while the neural models struggled with this task, the quantum model maintained similar scores to the other tasks. The non-linear neural model in particular consistently overfit the training data, with accuracy decaying to random guessing levels over the *Test* set. The quantum models seem to perform best on this task, as we see that the accuracy per stratum stays fairly flat over the *Test* set. The model accuracy per stratum is displayed in Figure 6.10c.

Though the overall results appear broadly similar (excepting for systematicity), we note that the neural architectures tended to require much more significant fine-tuning to obtain these results, while the quantum hyperparameters were more

robust across tasks.

6.2.1 Overgeneralisation

For overgeneralisation, we use a corrupted version of the productivity dataset, including contexts of up to 8 sentences in the training set. The validation set contained only uncorrupted examples, which we sampled in the *Vanilla* fashion from the entire *Train* distribution. We additionally evaluated the final models on an unseen, uncorrupted dataset drawn from the same distribution as the training set, described in Section 3.3.2 as *Productivity'* and labelled here as *Test'*.

As visualised in Figure 6.12a, the quantum model overgeneralises for corruption of up to 30%, and only starts significantly overfitting from 40% corruption. In Figure 6.12c, we see that the drop in accuracy from *Train* to *Test'* is largely proportional to the number of corrupted datapoints, though with a small factor. This confirms that the model is not able to overgeneralise perfectly, and may learn to overfit certain exceptions that then cause mistakes on the *Test'* set.

Meanwhile, the neural model (in this case a Flat model with Mish activation and wire dimension 12[1]) does not seem to exhibit as clear a trend, as per Figure 6.12b. In fact, the relatively consistent drop in accuracy from *Train* to the *Test'* dataset distribution in Figure 6.12d suggests that the model is likely overfitting many of the uncorrupted samples too, rather than learning the intended compositional rules. The model at 20% is particularly guilty of this, but even the reference model trained on an uncorrupted dataset exhibits a similar amount of decay as the model trained on a dataset that had 50% of the instances corrupted.

[1]We did not pursue the Linear model for this task due to time and resource constraints.

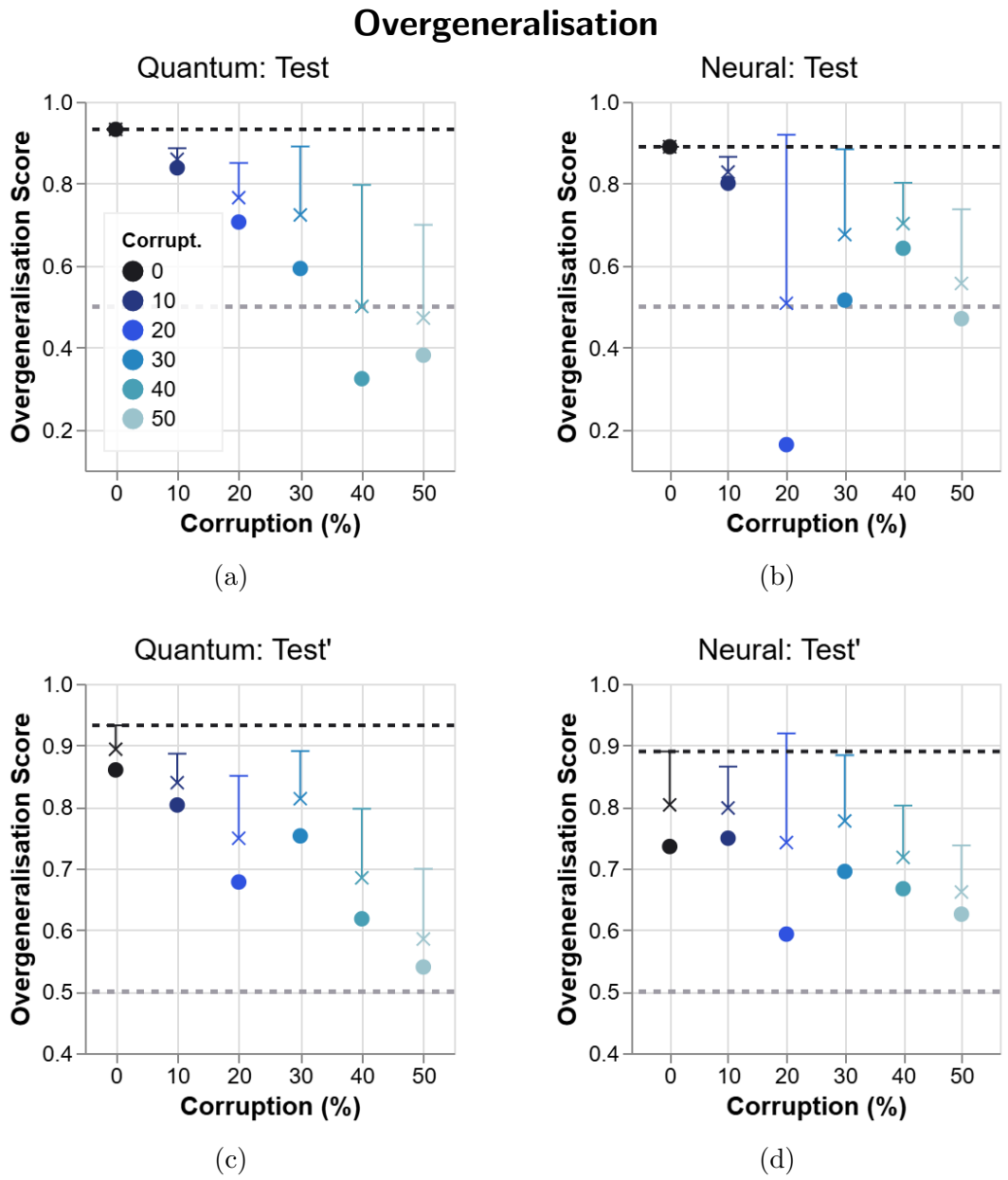


Figure 6.12: Overgeneralisation scores for (a,c) quantum and (b,d) neural models, trained on increasingly corrupted datasets (represented by the colour of the datapoints). We plot the overgeneralisation score as a filled point. The *Train* accuracy (over the uncorrupted training datapoints) is plotted as a dash. The *Test* or *Test'* accuracy is plotted with a cross. Plots (a) and (b) display the *Test* dataset composed of the *corrupted* datapoints seen during the model's training, in their uncorrupted form. Plots (c) and (d) consider the *Test'* dataset (drawn from the same distribution as the entire (uncorrupted) *Train* set). A line is shown between the *Train* and *Test* to highlight the magnitude of difference between the points. The *Train* accuracy of the models trained on the uncorrupted dataset (0%) is marked as a dashed black line for comparison, while the random guessing baseline of 50% is marked with a dashed grey line.

6.2.2 Training for compositionality

In this section, We consider how the validation split impacts model selection with regards to compositionality. Figure 6.13 displays the impact of the different validation choices on the selected model's final productivity score, as computed from the *Test* set.

We first notice that obtaining a high compositional score is sometimes at the cost of *Test* accuracy, as is the case with splits 0, 3 and 4. Similarly, a high *Train* score alone is insufficient to guarantee compositionality, as exhibited by splits 0, 2 and 3. This is by construction of the compositionality score - a high penalty is imposed on models that overfit the training distribution at the cost of ε-productivity.

This then seems to suggest one of the compositional schemes, or a scheme that encompasses both *Train* and *Test* distributions may be better. Both *Valid All* and *Valid AB* compositional schemes are unfortunately relatively unreliable estimators of the final compositionality score, however (see Figure 6.14 and associated commentary for a quantification). This may be due to the low number of samples available for validation (20% of the total dataset for each context depth sampled) leading to high volatility. Another source of error is from overfitting, where the model's high accuracy on the *Train* set heavily penalises the final compositionality score, but not the estimates computed from the validation sets.

Despite this, that the compositional score estimators tend to perform similarly to their accuracy based counterparts, but with a higher tendency to select models with lower overfitting between the *Train* and *Test* sets (as with splits 0 and 4). We hence suggest the compositionality score estimators (*Valid All* and *Valid AB*) as the best validation scheme candidates, with *Valid AB* being more optimised for efficiency, being a subset of the *Valid All* set and thus requiring fewer datapoint evaluations.

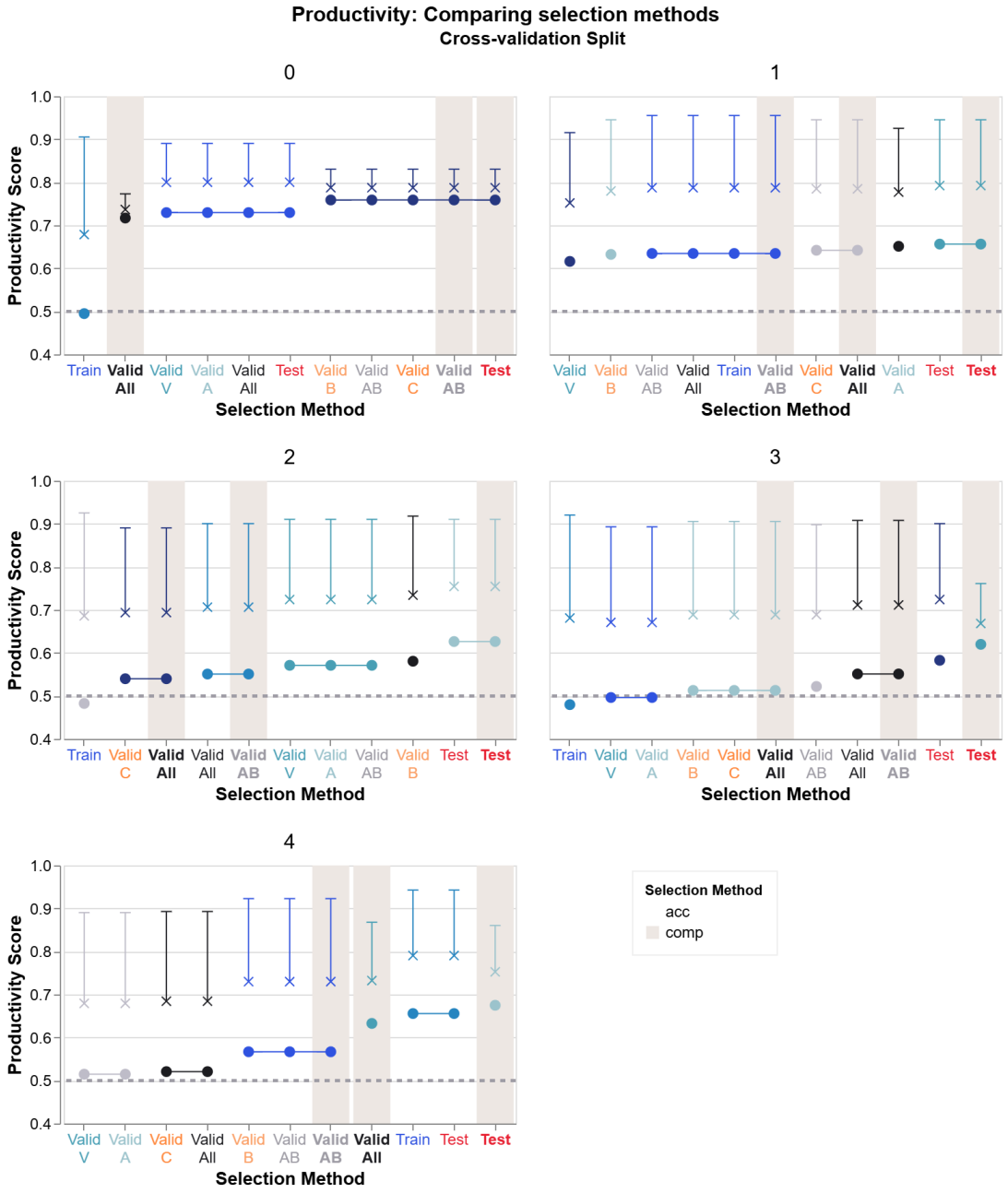


Figure 6.13: Productivity scores for the quantum models selected by each validation method, per cross-validation split. The productivity score is drawn as a point, with a bar displaying the difference in accuracy from *Train* to *Test*. The *Test* accuracy is highlighted with a cross, while *Train* is highlighted with a dash. In each case, the schemes are ranked, with the model whose productivity score was the best over the *Test* set added for reference. The plots are coloured according to the particular model selected per split. A line links the productivity scores obtained by the same model when selected by different schemes. The validation schemes that computed a proxy for the productivity score are highlighted with a shaded bar and bold text.

In Figure 6.14, we compare how closely each of the validation schemes track the actual productivity score of the model. As alluded to above, the *Comp All* and *Comp AB* schemes are very volatile, and exhibit quite a lot of noise when compared to the true compositionality score.

In about half of the cases (splits 0, 1 and 4) the best model selected by the schemes correspond to good models, if not the best available. These cases also correspond to instances where there was were fewer candidates with similar estimated scores. Indeed, the selection methods seem to work better when the actual score is above 0.60, as there tends to be a model nearby with a high estimated score that rises above the noisy scores of the middling-to-bad models.

Finally, compared to the *Valid V* accuracy, the compositionality score estimates select fewer candidate models. This is desirable as deciding between the candidates likely requires an expensive evaluation of a *Valid Comp* style dataset: though the best model, or one very close to it, is reliably within the top 4% of *Valid V* scores, there are so many other models with similar scores that picking it out from the noise would be inefficient.

This suggests that only extending the validation set may not be enough to guarantee that a good model is selected. However, using an estimator for the compositionality score as an initial selection method, followed by a secondary validation set (*Valid Comp*) to further refine the selection improves upon the naive compositional validation scheme used in Section 6.1. The results reported on in the rest of this work used the *Comp AB* scheme, with no further *Valid Comp* style selection due to time and resource constraints.

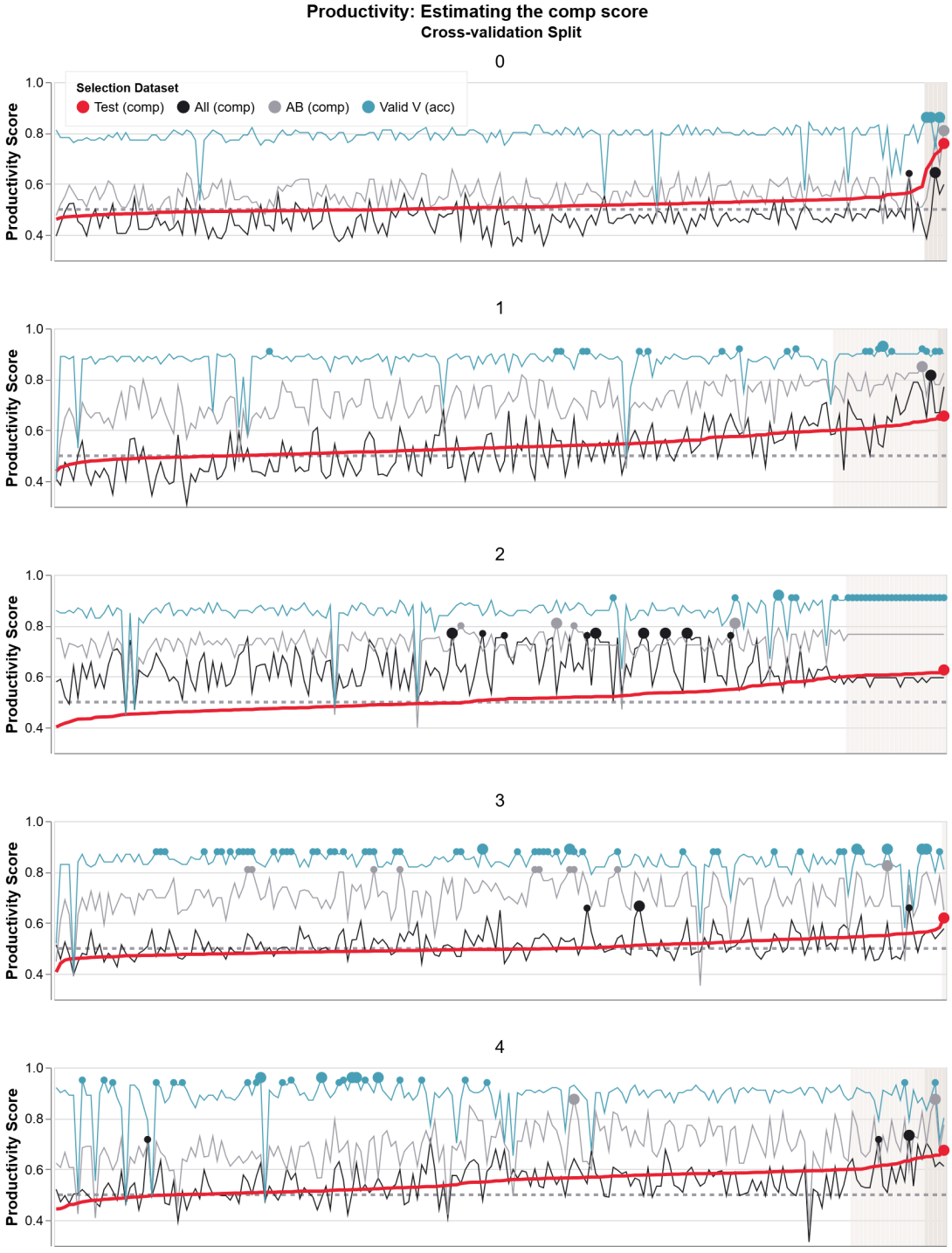


Figure 6.14: Comparing the validation and true compositionality scores for the models in each cross-validation split, for the quantum models. The true compositionality score is marked as a thick red line with the models arranged in increasing order. We plot the estimated compositionality score for each model as thin greyscale lines and the vanilla-style *Valid V* accuracy as a blue line. The top 4% of selected models (by score) are highlighted with a point, with the best models receiving a larger point. The areas where the true compositional score reaches 0.6 and 0.65 are shaded.

6.2.3 Cross-validation

For each of the architectures considered, we also performed 5-fold cross validation to explore how sensitive to the input data each method appeared. Figure 6.15 displays the productivity scores of the best models selected for each fold using the *Valid AB* comp selection method. The results are generally somewhat worse than plotted in Figure 6.9. This is likely due at least in part to the reduced size of the validation set (20% of the data available at each stratum rather than the 40% used when not cross-validating), leading to more a volatile compositionality score estimate. Additionally, for the quantum models, the maximum number of epochs used for each training run was reduced. Given that some of the selected models in the previous section were from epochs in the tail-end of their respective runs, it is possible that this earlier stopping caused some good candidates to be missed. To quantify how much the models overfit the *Train* data, we also evaluated them on a secondary sub-dataset of the productivity' dataset, drawn from the same distribution as the *Train* set and known as *Train'*.

The extended-validation quantum model shows the most variation between splits, which is also somewhat reflective of the available models for each split studied in Figure 6.14. Meanwhile, the quantum model trained with curriculum learning is much more uniform, though unfortunately at the cost of selecting models with compositionality scores below 0.6.

Looking towards the neural models, we see that both perform similarly, though the linear models show more substantial overfitting than the non-linear models: indeed, when considering the compositionality score calculated from the *Train'* distribution, both neural architectures obtain very similar scores.

In Figure 6.16, a more detailed breakdown of the model performance per training stratum (context depth) is shown over the *Train* and *Test* datasets.

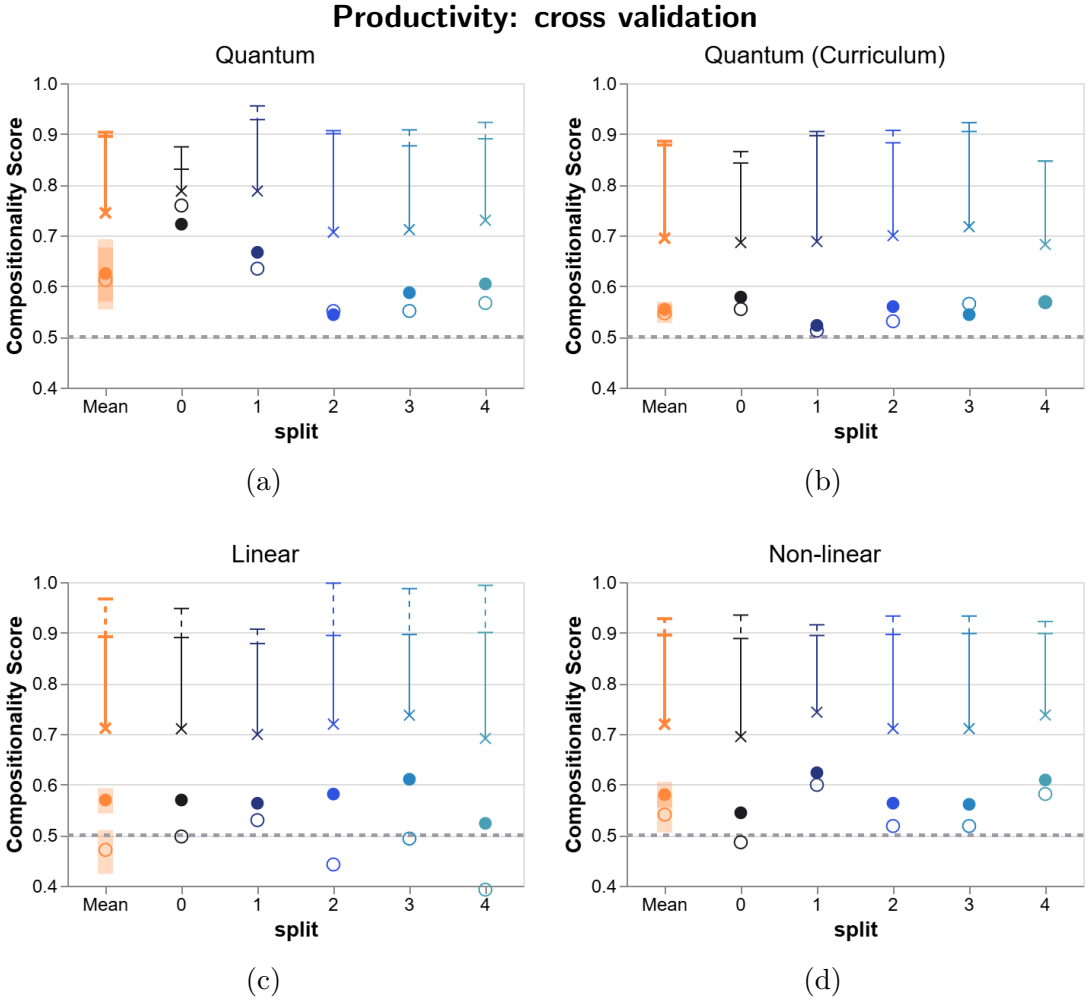


Figure 6.15: Best selected models according to the *Comp AB* scheme, and the mean performance across the five splits, for the various model architectures considered. (a) and (b) display the quantum architecture (Sim4 with 3 layers and 1 qubit per noun), where (b) has been trained via the curriculum method. (c) and (d) display the neural architectures considered: Linear (36 dimensions per noun) and Flat (12 dimensions per noun and Mish activation). The productivity score is plotted as a circle for each split. The mean value is shown in orange, with shaded confidence interval. The *Train* and *Test* accuracies used to compute the productivity score are plotted as a dash and cross respectively, and linked by a line. The dashed line indicates the drop in accuracy between the *Train* set and the *Train'* set. A filled circle indicates the compositionality score relative to the *Train'* distribution, while the open circle plots the score relative to the *Train* accuracy.

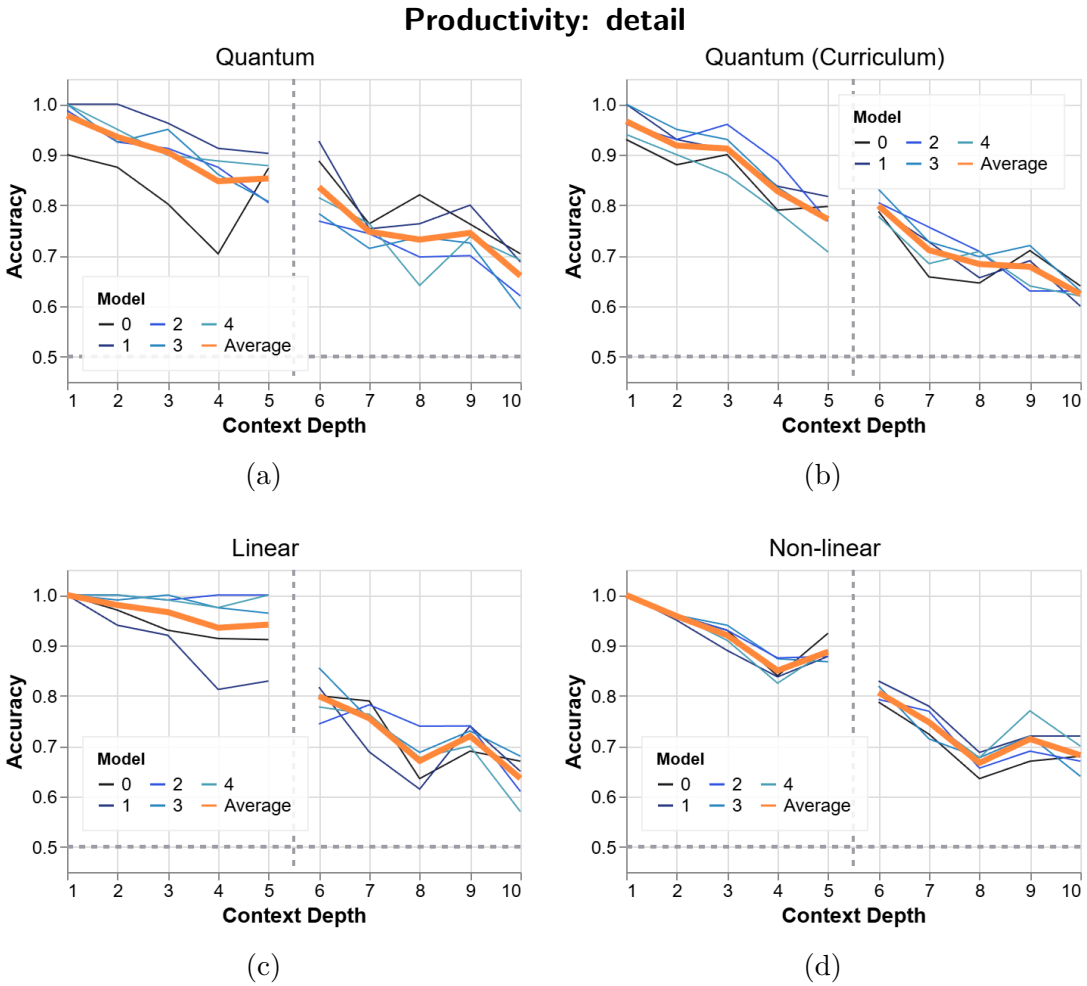


Figure 6.16: Visualising the model performance as a function of context depth for each split. (a) and (b) display the quantum architecture (Sim4 with 3 layers and 1 qubit per noun), where (b) has been trained via the curriculum method. (c) and (d) display the neural architectures considered: Linear (36 dimensions per noun) and Flat (12 dimensions per noun and Mish activation). The mean in a thick orange line. The dashed horizontal line represents the random guessing baseline accuracy, and the vertical line denotes the split between the *Train* and *Test* datasets.



CHAPTER 7

Compositional understandability

While a trained model that succeeds on its testing set is a useful outcome, we would typically want some further guarantees that the model is doing what we would like it to before trusting its output, especially in critical situations. For the classes of models studied here - which are syntacically compositional - we can take this extra step and analyse them further to extract better knowledge of what they are doing. In this chapter, we select two models to analyse in detail, considering interpretation techniques that could remain applicable when applied to larger-scale models.

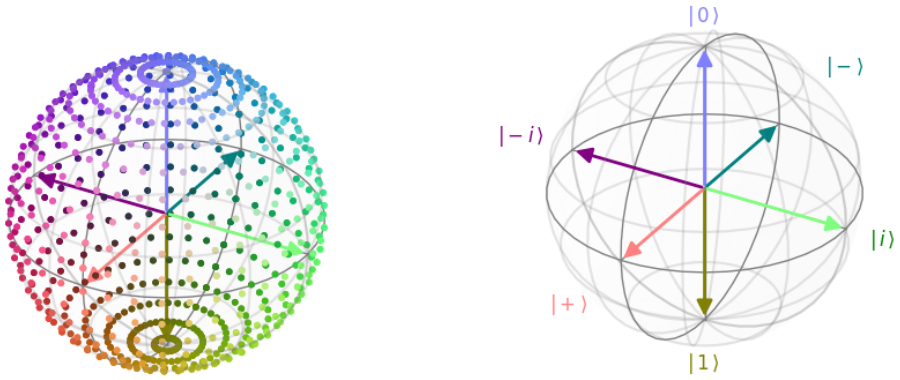
The built-in compositional structure of DisCoCirc allows us to interpret its behaviour via *compositional understandability*. We begin by studying how each learnt component of the model behaves individually. Then, we use the model's inherent compositional structure to piece together how it performs on each respective dataset. We extend our analysis by examining certain diagram fragments, and compare how they relate to one another in order to find underlying rules the model has learnt to apply.

We first study the models trained on the *following* datasets in Section 7.1, studying the model's behaviour both *mechanistically* - by directly interpreting the black boxes of the model - and *compositionally*, by considering the behaviour of diagram fragments. In Section 7.1.2, we observe that the two-directional model behaves similarly to an ideal model we construct by hand. In Section 7.1.3, we explain how the four-directional model differs from the ideal model, performing well nonetheless. For it to correspond to an ideal model exactly, the four-directional model would require two qubits per wire instead of our choice of one qubit per wire (made due to hardware constraints).

The models studied in this work allow for direct visualisations on the Bloch sphere, which enables the mechanistic interpretability in Section 7.1, however this is only feasible due to the small-scale nature of the models. The diagram fragment based techniques used can nonetheless be extended to situations where such a representation is not possible (for example in a model that assigns multiple qubits per noun wire). We demonstrate this more general approach in Section 7.2, using it to explain how the quantum model trained on the productivity dataset works on a compositional rather than mechanistic level. We find that it instead learns to answer a more general question than intended, which accounts for the decay in accuracy as the stories get longer.

7.1 Following datasets

Throughout this section, we make use of the following representations to visualise the quantum components of the model:



(a) Reference states $|\psi\rangle$ (b) Reference points $|\phi\rangle$

Figure 7.1: Reference colours. (a) Colour mapping used for paired Bloch sphere pure states $|\psi\rangle$. The reference points $|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle$ are highlighted as vectors, and displayed separately in (b).

Single-qubit states These are visualised on a Bloch sphere in the standard way, where pure states lie on the surface of the sphere, and mixed states lie inside the sphere.

Two-qubit states Following Altepeter et al. (2009), two-qubit states $\rho_{1,2}$ are visualised as coloured ellipsoid surfaces inside a pair of Bloch spheres in terms of partial projections. In this representation, each sphere corresponds to

one of the qubits. The first Bloch sphere displays a surface defined by $\rho_{1,\psi} = (\mathbb{I} \otimes \langle\psi|)\rho_{1,2}(\mathbb{I} \otimes |\psi\rangle)$ for all pure states $|\psi\rangle$ that the second qubit can be in. Each state $\rho_{1,\psi}$ is coloured as per the reference colour of $|\psi\rangle$, shown in Figure 7.1a. Analogously, the second Bloch sphere displays a surface describing the state $\rho_{\psi,2}$ of the second qubit given that the first qubit is in state $|\psi\rangle$.

Single-qubit gates We visualise single-qubit gates as rotations on the Bloch sphere, plotting the trajectory of reference points $|\phi\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle\}$, under the action of the rotation U. The final state $U|\phi\rangle$ is coloured according to the colour of the initial state $|\phi\rangle$, as given in Figure 7.1b.

7.1.1 Hand-crafted Clifford models

Given the simplicity of the task, we are able to construct a perfect, deterministic, compositional model that solves the task by hand. The two-directional model is described in Figure 7.2, while Figure 7.3 describes the four-directional model. These models are constructed to perfectly implement the axioms required to solve the task. In the two-directional case, this provides a target for the trained model to strive towards, as well as a baseline for comparison. Figure 7.4, Figure 7.5 and Figure 7.6 visualise the two-directional model.

Two-directional

The model associates $|\texttt{North}\rangle$ with $|0\rangle$ and $|\texttt{South}\rangle$ with $|1\rangle$. Turning around is implemented as a bit-flip. `follows` discards the first qubit, then copies the second qubit along the computational basis, such that the state of the first qubit will match that of the second when projected onto the computational basis. Finally, the assertions become bell states checking for correlation or anti-correlation along the computational basis.

Four-directional

Here, each of the cardinal directions is assigned a state of the computational basis: $|\texttt{North}\rangle = |0,0\rangle$, $|\texttt{South}\rangle = |1,0\rangle$, $|\texttt{East}\rangle = |0,1\rangle$, $|\texttt{West}\rangle = |1,1\rangle$. We can interpret these states as numbers, such that the rotations can be implemented as addition (modulo 4): `left` acts as $+1$, `around` as $+2$ and `right` as $-1 \equiv +3$. As with the two-directional model, the `follows` gate discards the wires corresponding to the first actor, then copies the state of the second noun along the computation

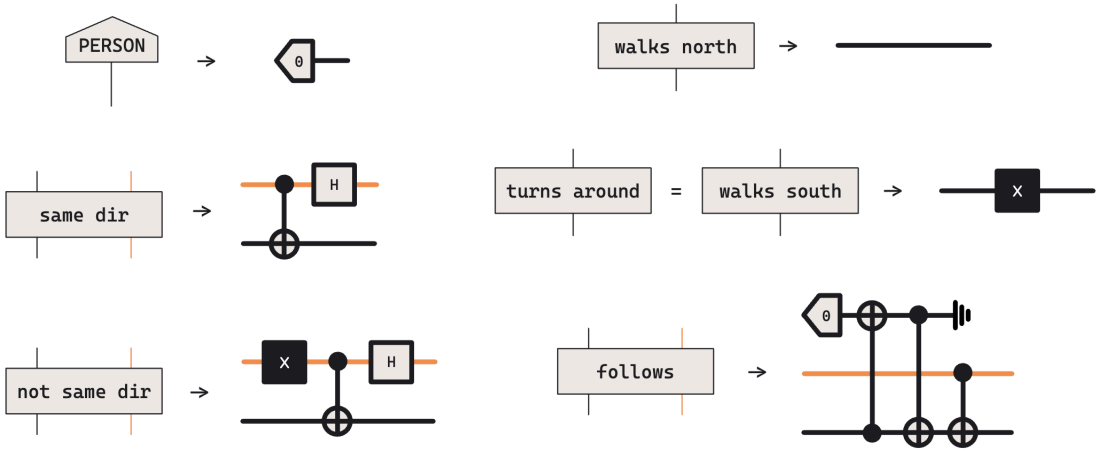


Figure 7.2: An implementation of a deterministic model for the two-directional dataset. For two-actor gates, the second actor is coloured as a visual aid.

basis. The positive assertion `same dir` acts as a bell state, so the overlap will be one when the states are the same and zero otherwise, whilst the overlap with `not same dir` will always be one half, and thus the maximum of the two checks if the states are the same or different.

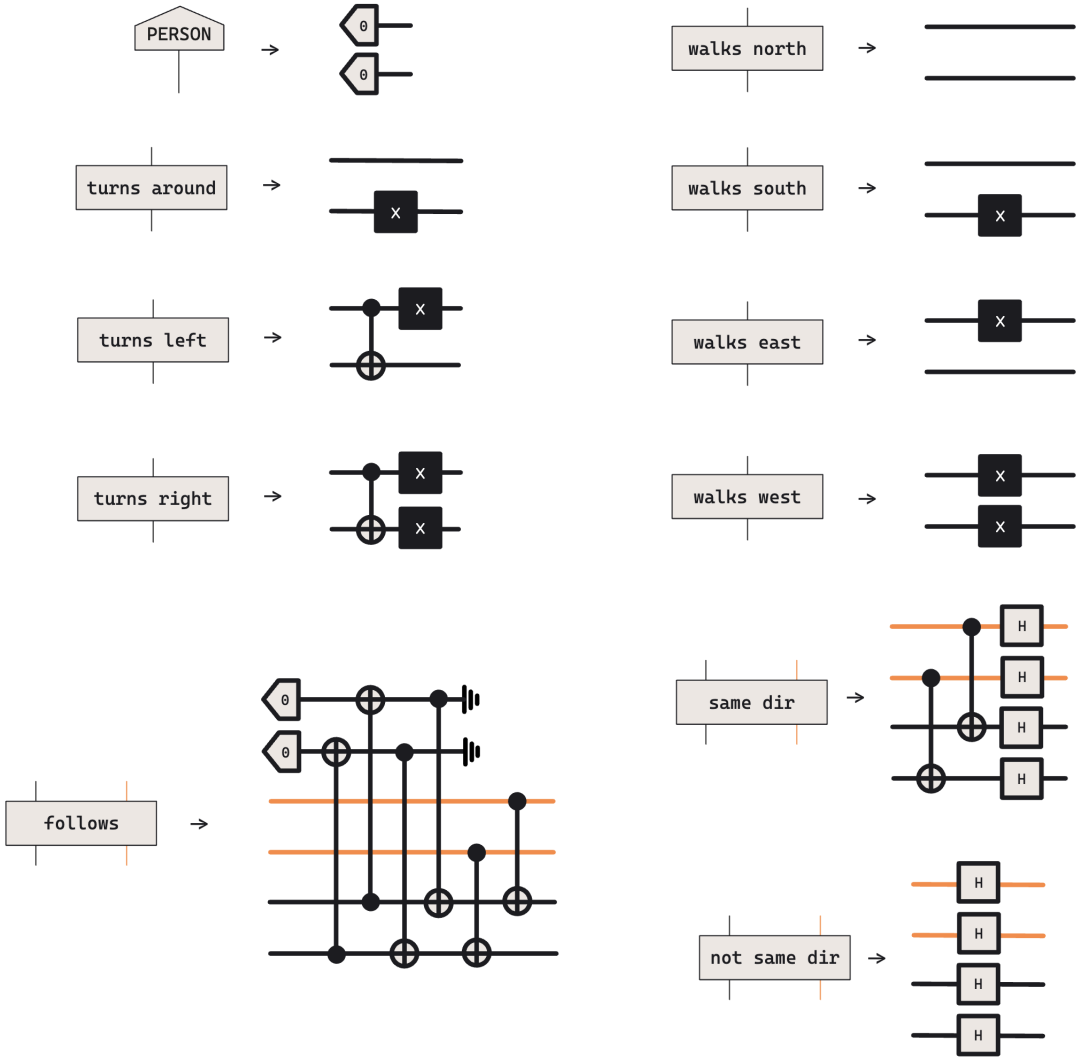
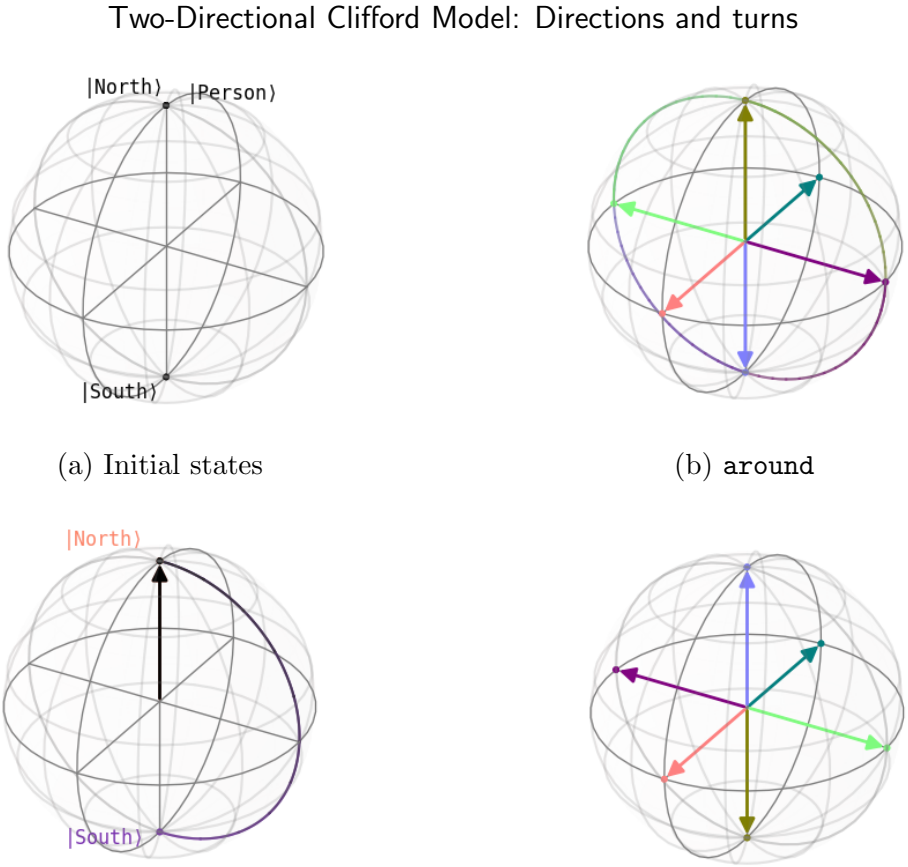


Figure 7.3: An implementation of a deterministic model for the four-directional dataset. For two-actor gates, the qubits corresponding to the second actor are coloured as a visual aid.



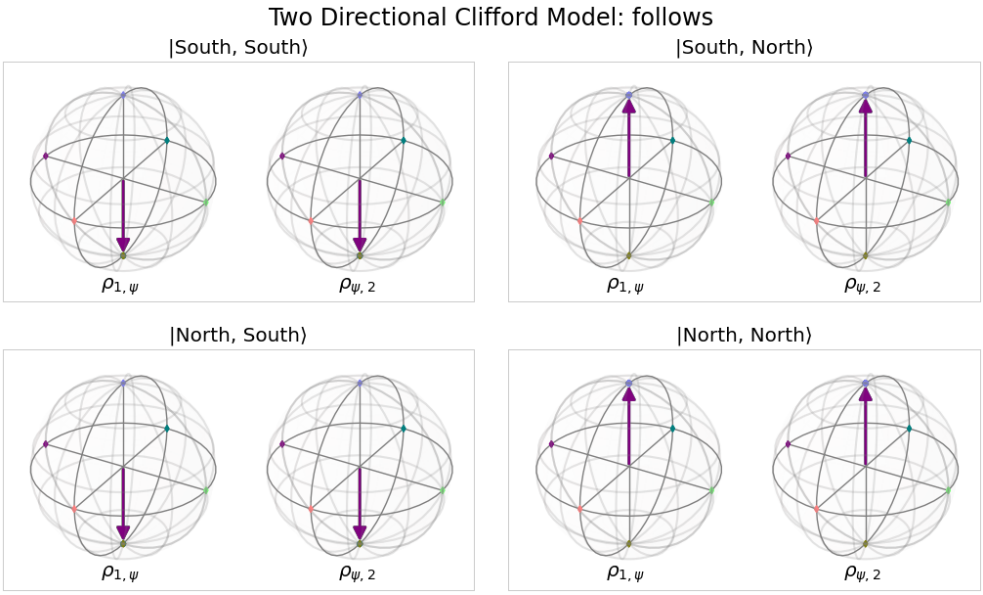
(a) Initial states

(b) `around`

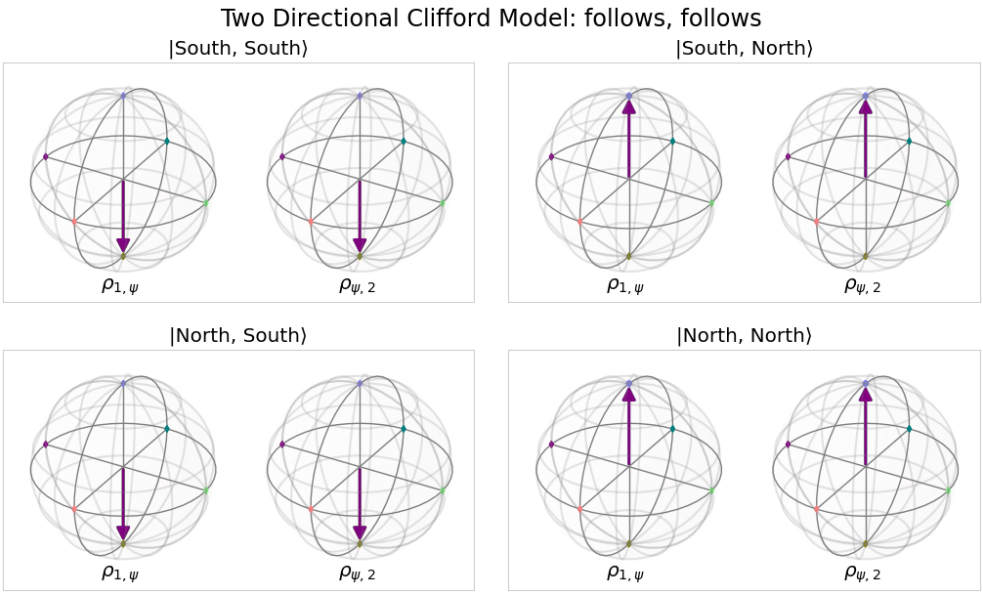
(c) $|\texttt{North}\rangle \approx U(\theta_{\texttt{around}})\,|\texttt{South}\rangle$

(d) $U(\theta_{\texttt{around}})^2 = Id$

Figure 7.4: Visualising the two-directional Clifford model. (a) Displays the direction states, while (b) shows the action of the `turns around` gate. (c) and (d) verify that the expected single qubit axioms hold.

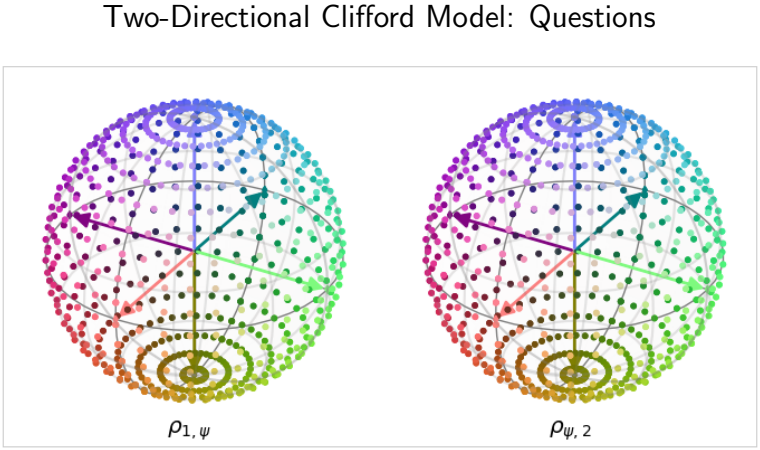


(a)

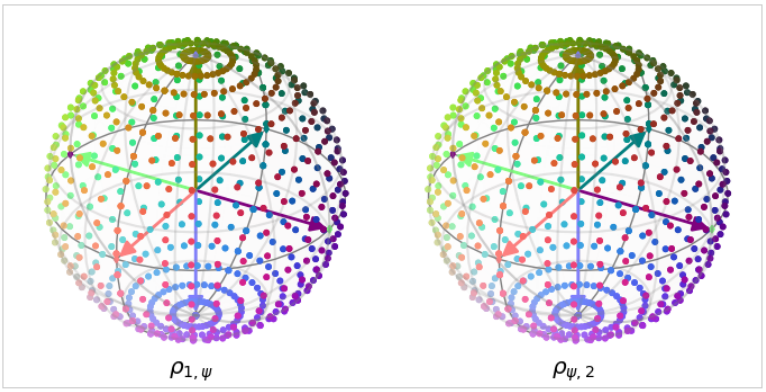


(b)

Figure 7.5: Visualising (a) a single application, and (b) two successive applications of the `follows` gate for the two-directional Clifford model. All states are separable, such that the paired Bloch spheres can be read like single qubit Bloch spheres describing each qubit separately. The plots in (a) and (b) are exact matches, as the model captures the axiom perfectly. Note that the reference states (from Figure 7.1b) are plotted as diamonds for reference.



(a) *goes in the same direction*



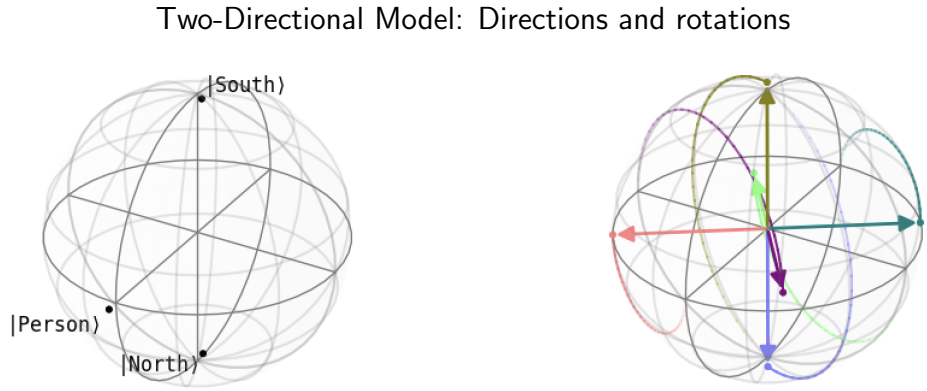
(b) *does not go in the same direction*

Figure 7.6: Visualising the (a) positive and (b) negative assertion states for the two directional Clifford model. Both states are maximally entangled pure states, corresponding to the Bell basis states $|\texttt{North},\texttt{North}\rangle + |\texttt{South},\texttt{South}\rangle$ and $|\texttt{North},\texttt{South}\rangle + |\texttt{South},\texttt{North}\rangle$ respectively.

7.1.2 Trained model: two-directional dataset

Initial States

Since these only ever occur at the start of a text, they can be represented as states (rather than rotations). Note that the state labelled $|\texttt{North}\rangle$ represents the state obtained from the entire sentence '*PERSON walks north*', and similarly for $|\texttt{South}\rangle$, as this is how it is always encountered in the text. These are depicted in Figure 7.7a.



(a) Initial states

(b) `around`

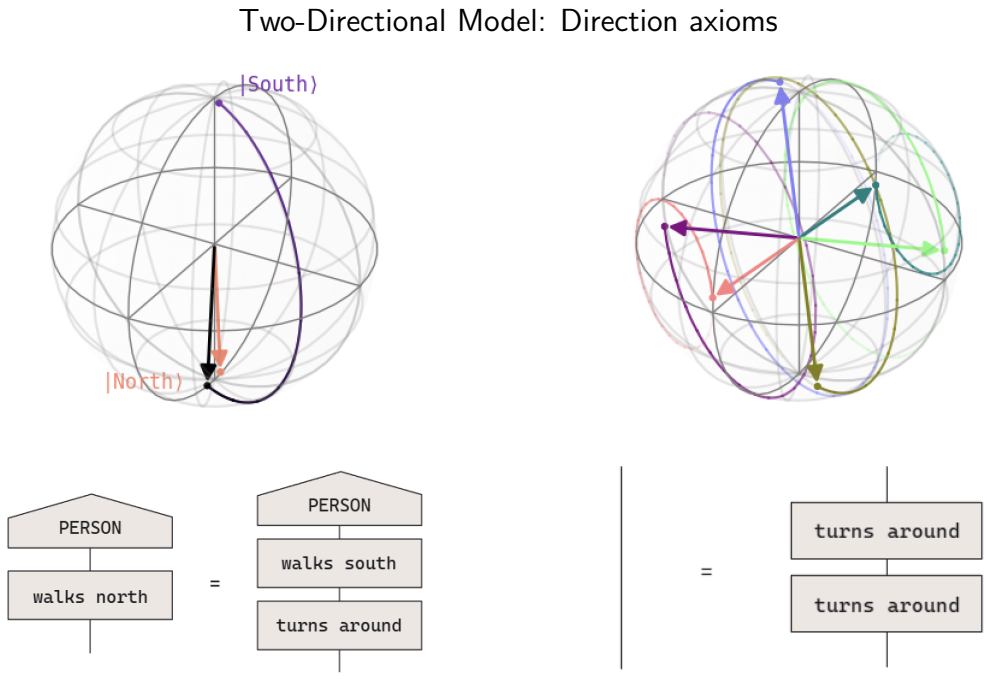
(c) $U(\theta_{\texttt{around}})^{30}$

Figure 7.7: Single qubit gate analysis for the two directions model. (a) Single qubit initialisations, viewed as states. The two directions are aligned with the computational basis. (b) The effect of the single qubit rotation `turns around` on the reference points. (c) Applying `turns around` 30 times resembles applying it just once.

Single-qubit gates

In Figure 7.7b, we plot the single-qubit rotation `turns around`. The gate is approximately a π rotation about the x-axis (up to a small phase). Since the

directions $|\texttt{North}\rangle$ and $|\texttt{South}\rangle$ are approximately on the poles of the sphere, the expected axioms also hold approximately (Figure 7.8). Since the rotation is only *approximately* π, there is a small offset that can accumulate if `turns around` is applied successively - Figure 7.7c shows the effect of applying `turns around` thirty times in a row, demonstrating that an extra π rotation has been accumulated: thirty applications resemble a single application, whilst we would expect that turning around an even number of times should be equivalent to not turning around at all. A perfect model (as described in Figure 7.2) would not exhibit such behaviour, however the model has learnt to accommodate this error, as will be shown in the next section.



(a) $|\texttt{North}\rangle \approx U(\theta_{\texttt{around}})\,|\texttt{South}\rangle$ (b) $U(\theta_{\texttt{around}})^2 \approx Id$

Figure 7.8: Single qubit axioms for the two-directional dataset. We verify that the two-directional model satisfies them approximately: (a) The final states (highlighted as vectors) are very close to one another. Note that in this case the colours are to be taken as a visual aid only. (b) We see that the reference points are almost mapped back to their original positions. The slight discrepancy is what leads to the peirodicity shown in Figure 7.7c.

Two-qubit gates

As our ansatz contains some built-in semantic rewrites, there is only one two-qubit gate: `follows`. We consider its action on a set of states defined by the possible

directions the two people involved might be initially facing. Since $|\texttt{North}\rangle$ and $|\texttt{South}\rangle$ are approximately orthogonal, this acts as a basis, and so investigating the action of `follows` on each of these initial states fully characterises the gate. Depicted in Figure 7.9, the states considered encode texts of the form:

$$\text{`}A \textit{ walks } x. \; B \textit{ walks } y. \; A \textit{ follows } B.\text{'} \tag{7.1}$$

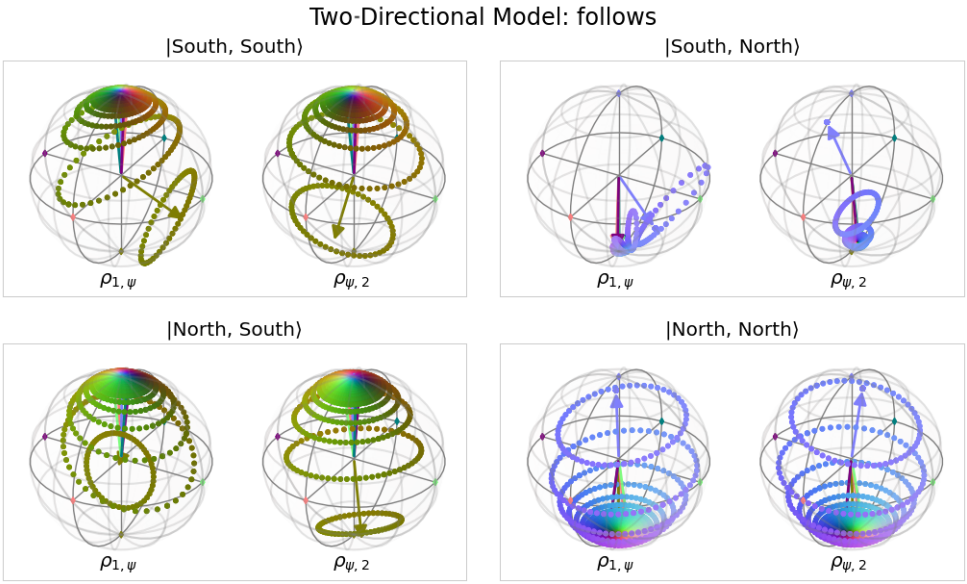
for $x, y \in \{North, South\}$, and A and B corresponding to qubit 1 and 2 respectively. The gate behaves largely as a joint projection of the qubits onto the $|\texttt{North}\rangle$ - $|\texttt{South}\rangle$ axis, controlled by the state of qubit 2. The joint aspect of this projection entangles the qubits in such a way that some certainty as to the original directions faced is lost, however since the state of the A remains correlated with the new location of B, and largely independent of the original direction A was facing, the `follows` gate seems to have correctly captured the notion of following.

We also visually confirm that the axiom

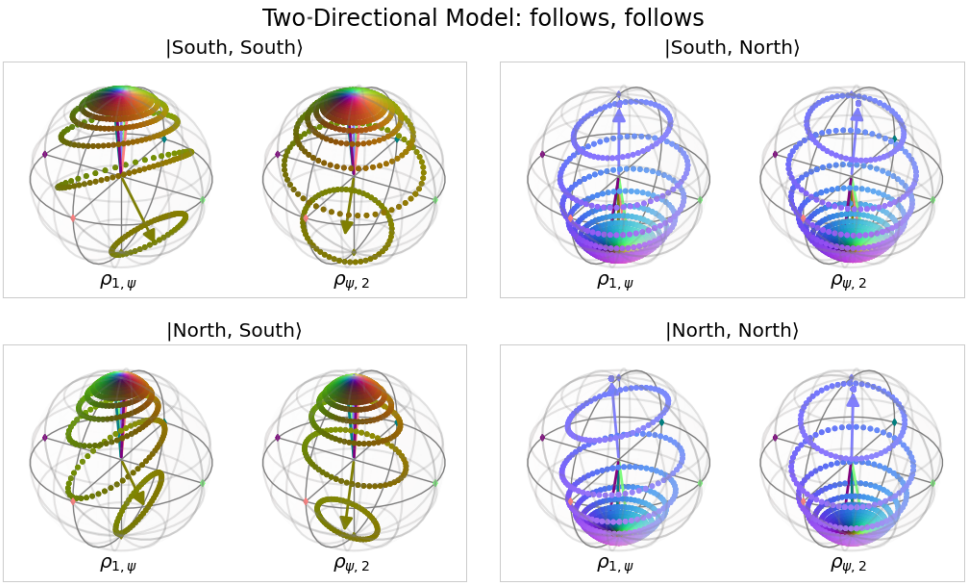
$$\text{`}A \textit{ follows } B. \; A \textit{ follows } B.\text{'} = \text{`}A \textit{ follows } B.\text{'} \tag{7.2}$$

approximately holds, by comparing the visualisations for both gates. Applying following a second time induces a slight phase shift, but the state remains at the relevant pole with high probability.

We also note that the projective aspect of `follows` can correct for some accumulated errors from `turns around`. Although we would expect even powers of `turns around` to be approximately the identity, we find that $U^{30}(\theta_{\texttt{around}}) \approx U(\theta_{\texttt{around}})$, as demonstrated in Figure 7.7c. Provided `follows` is interspersed between `turns around` sufficiently frequently, the states will be projected back towards the pole they are closest to, thus clearing the error accumulated so far. Sufficiently frequently in this case means at most 15 `turns around` gates have been applied in succession, as from this point the probability of projecting to the correct state will dip below 50%.



(a)



(b)

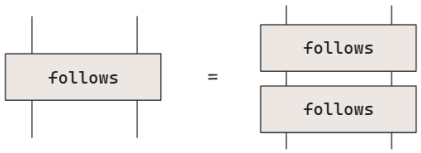


Figure 7.9: Visualising the two-qubit `follows` gate for the two-directional model to confirm axiom (7.2). The label $|x, y\rangle$ identifies the state $U(\theta_{\texttt{follows}})\,|x, y\rangle$. The expected joint final state is $|y, y\rangle$. Note that we mark the reference points with a coloured dot on the paired Bloch spheres as a visual aid; these are not to be considered part of the plotted ellipsoids. a. Single application of the `follows` gate. b. `follows` applied twice.

Figure 7.9 (previous page): Reading the plot: a. Consider the paired Bloch sphere for $|\texttt{South, South}\rangle$ for a single application of `follows`. The left-most sphere shows that $\rho_{1,\psi} \approx |0\rangle \approx |\texttt{South}\rangle$ for $|\psi\rangle \not\approx |1\rangle \approx |\texttt{North}\rangle$, and vice-versa for $\rho_{\psi,2}$. This suggests that the pair of qubits is very close to being in the $|00\rangle \approx |\texttt{South, South}\rangle$ state. Although somewhat mixed, many of the points are close to the surface of the sphere, suggesting that the qubits are entangled (though not maximally so). This aligns with the interpretation that `follows` acts as an approximate projection of the first qubit onto the state of the second (as measured in the $\{|\texttt{North}\rangle, |\texttt{South}\rangle\}$ basis), with an additional entangling component that allows for $\rho_{1,\texttt{North}} \approx |\texttt{North}\rangle$ (whereas we would expect a perfect projection to give $\rho'_{1,\texttt{North}} \approx |\texttt{South}\rangle$). The paired state for $|\texttt{North, South}\rangle$ is very similar, whilst $|\texttt{North, North}\rangle$ is instead concentrated towards the $|11\rangle \approx |\texttt{North, North}\rangle$ state. For $|\texttt{South, North}\rangle$, we can see the state as being even more concentrated towards $|\texttt{North, North}\rangle$, to the point where the state is almost separable along this basis. b. The paired Bloch spheres describing the application of `follows` twice are visually quite similar to those describing `follows` once, however the colours are rotated slightly along the vertical axis, representing an induced phase shift. In the $|\texttt{South, North}\rangle$ case, there is a more significant difference, as the second application of `follows` results in a more entangled state, though in both cases the final state remains approximately $|\texttt{North, North}\rangle$. We thus conclude that axiom (7.2) holds at least approximately.

Questions

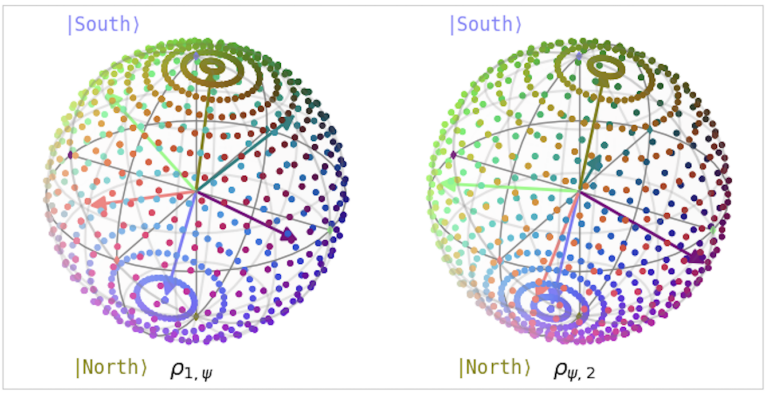
Both assertion effects are maximally entangled, approximately corresponding to two states from the Bell basis. Written in terms of our initial states, we have:

$$|\texttt{same dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{North}\rangle \otimes |\texttt{North}\rangle + |\texttt{South}\rangle \otimes |\texttt{South}\rangle)$$

$$|\texttt{opp dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{South}\rangle \otimes |\texttt{North}\rangle + |\texttt{North}\rangle \otimes |\texttt{South}\rangle)$$

This intuitively captures the meaning of these questions (see Figure 7.10 for visualisations).

(a) *goes in same direction*



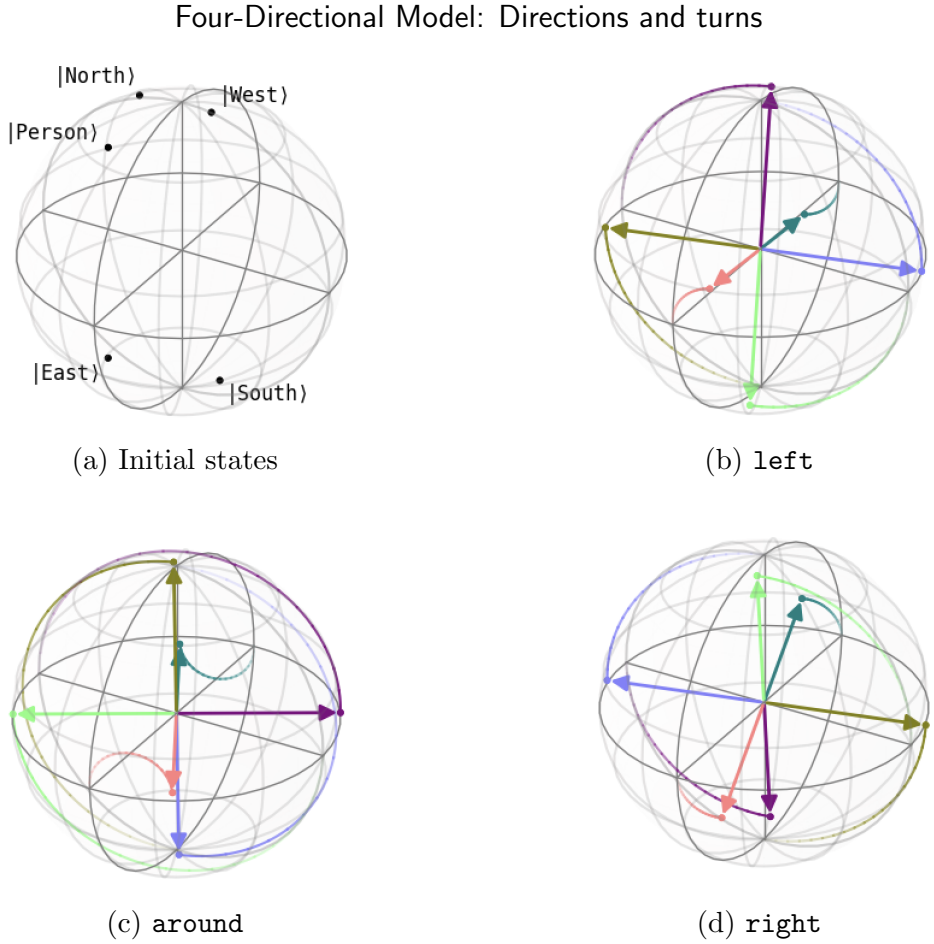
(b) *does not go in the same direction*

Figure 7.10: Visualising the assertions for the two-directional model. The main directions distinguished by the model are labelled and coloured according to the state identified. These are all maximally entangled pure states as both paired surfaces extend over their entire spheres. (a) `same dir`: we see here that $\rho_{1,\psi} \approx \rho_{\psi,2}$, such that states are correlated along |`North, North`⟩ and |`South, South`⟩ with no phase shift. We also see that the possible states $\rho_{1,\psi}, \rho_{\psi,2}$ are slightly concentrated towards the $|0\rangle$ state, which suggests the model can tolerate more error when the final state of the text *should* be |`South, South`⟩, but has instead deviated towards the equator. (b) `not same dir`: Conversely, the negative question expresses an anti-correlation along |`North`⟩ - |`South`⟩, with a slight phase shift between the two surfaces. $\rho_{1,\psi}$ and $\rho_{\psi,2}$ also exhibit a slight skew towards |`South`⟩ and |`North`⟩ respectively, which would allow the model to tolerate more error when the final text state is expected to be |`South, North`⟩.

7.1.3 Trained model: four-directional dataset

Initial states

For four directions, we have two additional inital states, |`East`⟩ and |`West`⟩, shown in Figure 7.11a. Like the two-directional model, this model places |`North`⟩ and |`South`⟩ at approximately orthogonal locations. |`East`⟩ and |`West`⟩ are similarly orthogonal, however these two 'bases' are not maximally distinct from each other: |`North`⟩ is somewhat close to |`West`⟩, and |`South`⟩ and |`East`⟩ are similar.



(a) Initial states

(b) `left`

(c) `around`

(d) `right`

Figure 7.11: Visualising single qubit gates for the four-directional model: (a) Single qubit initialisations. (b) Single qubit rotation `turns left` - the other rotations are defined in terms of `turns left`. (c), (d) Visualising the derived single qubit gates for the four-directional model.

Single-qubit gates

Though there are three single qubit gates in the text, the semantic rewrites built into the ansatz express each of these in terms of `turns left` only. `turns left` is

approximately a $\pi/2$ rotation about the x-axis as shown in in Figure 7.11b, so we can derive that `turns around` is approximately a π rotation, and `turns right` a $3\pi/2$ rotation about the same axis. The derived turns are depicted in Figure 7.11.

The single-qubit axioms hold less well in the four directions model, due to the fact that the directions are clustered into pairs. Figure 7.12 shows that `turns left` has the expected period of (approximately) 4. The axioms concerning `turns around` mostly hold, but not the axiom establishing the quarter-turns between the initial directions, as per Figure 7.13. The favouring of `turns around` can be explained by its featuring in the decomposition of `goes in the opposite direction`, as this will bias the dataset towards the occurrence of `turns around` compared to `turns left` or `turns right`. Additionally, the denser datasets have fewer turns, such that the `turns around` gates occurring as part of `goes in the opposite direction` will dominate the distribution. As such, the optimisation will tend to favour parameters where `turns around` interacts correctly since this will have a greater impact on the accuracy of the model.

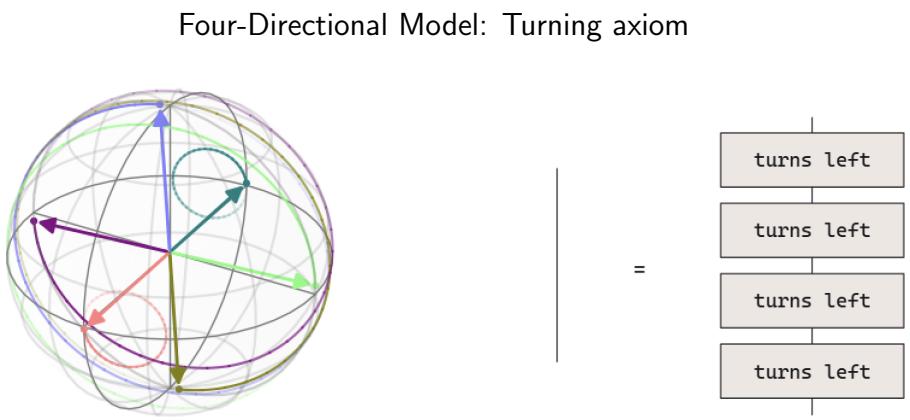
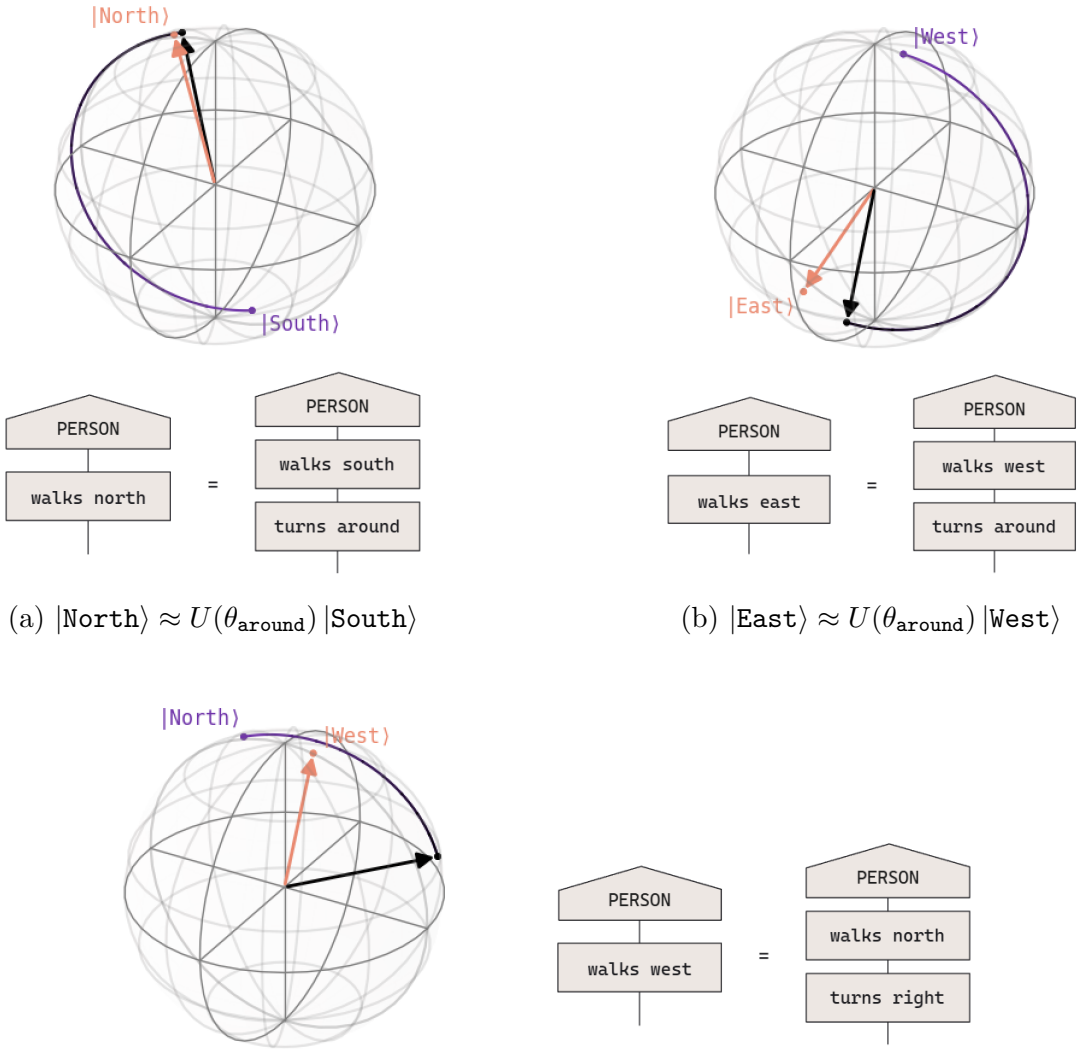


Figure 7.12: $U(\theta_{\texttt{left}})^4 \approx Id$: `turns left` four times is (approximately) the identity.



(a) $|\texttt{North}\rangle \approx U(\theta_{\texttt{around}})\,|\texttt{South}\rangle$

(b) $|\texttt{East}\rangle \approx U(\theta_{\texttt{around}})\,|\texttt{West}\rangle$

(c) $|\texttt{West}\rangle \neq U(\theta_{\texttt{left}})\,|\texttt{North}\rangle$

Figure 7.13: Single qubit axioms for the four-directional dataset. (a), (b) the axioms relating $|\texttt{North}\rangle$ to $|\texttt{South}\rangle$, and $|\texttt{East}\rangle$ to $|\texttt{West}\rangle$ hold approximately. (c) The model fails to satisfy the final axiom relating $|\texttt{North}\rangle$ to $|\texttt{west}\rangle$. Note that the initial states are coloured as a visual aid only.

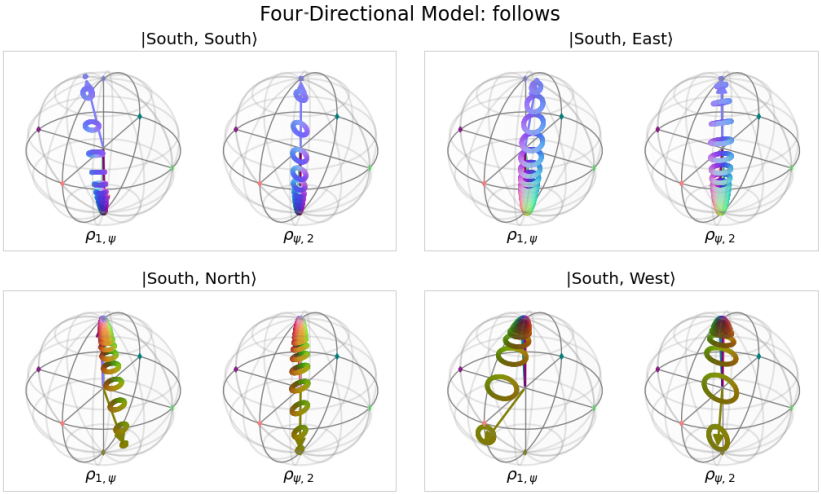
Two-qubit gates

We have only one two qubit gate to consider: `follows`. We additionally consider initial states including the two new directions. Like the two directions model, the four directions model projects the state of A (qubit 1) onto a state matching B (qubit 2) independent of the initial state of A, this time with significantly less entanglement. Since such a projection can only occur along a single axis, we see an even clearer pairing of the directions, such that the model is effectively collapsing into a two-directions model over states we shall call `North-West` ($|\texttt{NW}\rangle$) and `South-East` ($|\texttt{SE}\rangle$).

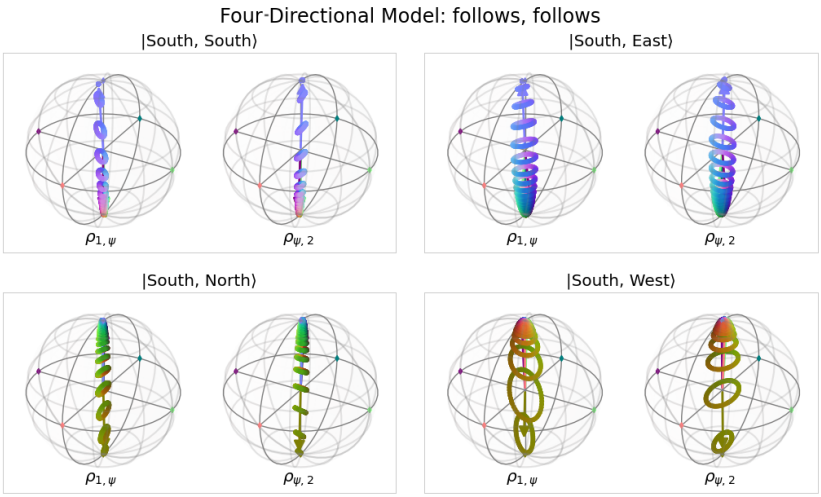
Figure 7.14a visualises the following gate on a subset of the 16 initial states. We visually confirm the approximate idempotence of following in Figure 7.14b. Finally, we exhibit an axiom the model fails to adequately capture in Figure 7.15:

$$\text{`}B \textit{ turns left. } A \textit{ follows } B.\text{'} = \text{`}A \textit{ follows } B. \textit{ } B \textit{ turns left. } A \textit{ turns left.}\text{'} \tag{7.3}$$

This axiom captures the idea that turns can be copied through `follows` when they act on the person being followed, however the model is unable to handle the quarter turn correctly.



(a)



(b)

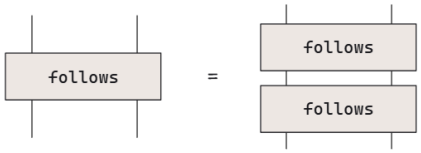
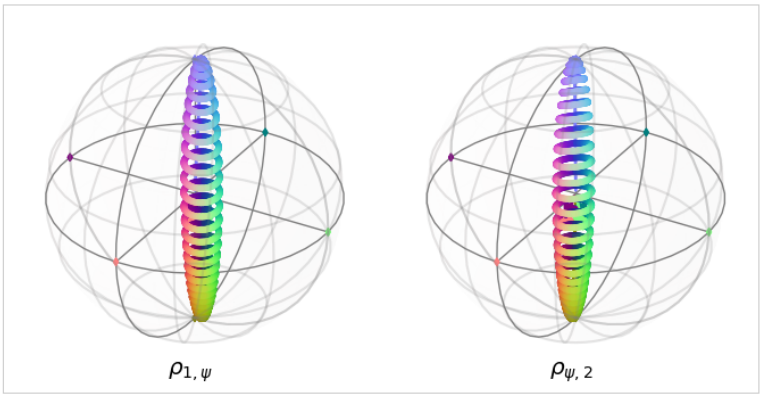
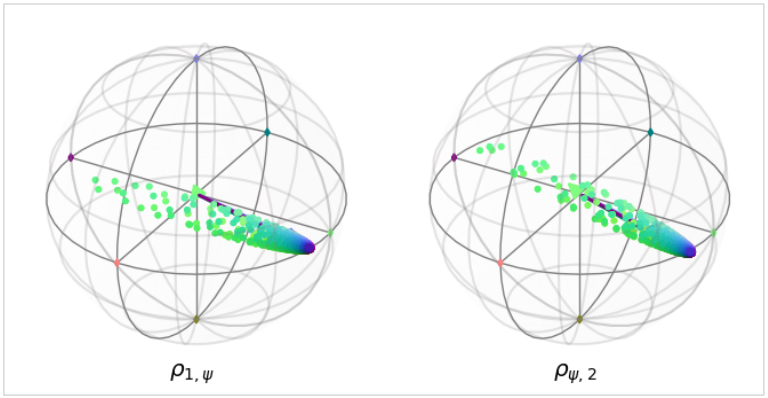


Figure 7.14: Visualising the effect of the two qubit `follows` gate of the four-directional model. For brevity, we consider its application to each of the possible initialisations for the second actor, fixing the first actor to always walk south. See Figure 7.17 for further plots. The label $|x, y\rangle$ identifies the state $U(\theta_{\texttt{follows}})\,|x, y\rangle$. The expected joint final state is $|y, y\rangle$. The paired Bloch spheres for the four directional model are very similar to the two directional model, except that the surfaces are much thinner. This suggests that the `follows` gate is acting much more like a projection, keeping the final states relatively un-entangled (though they remain classically correlated). We verify the (approximate) idempotence of the two qubit `follows` gate. Comparing the result of applying `follows` twice (b), to applying `follows` only once (a), we see that a phase shift is applied, though the shape of the states are very similar.

Four-Directional Model: Failed axiom



(a) '*A turns left.A follows B.*'



(b) '*A follows B. B turns left. A turns left.*'

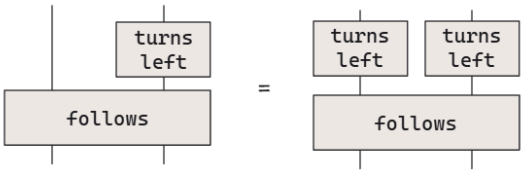
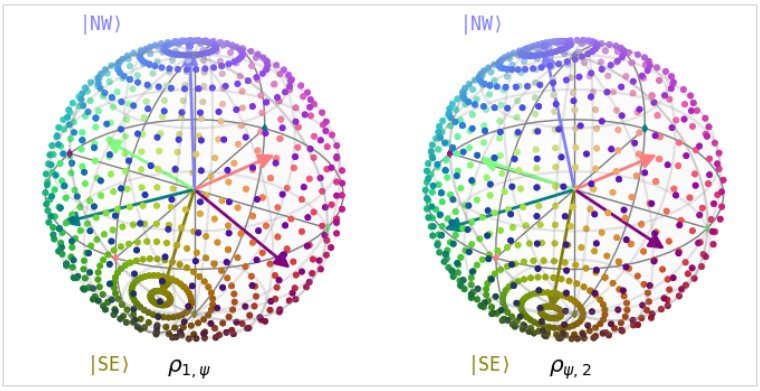
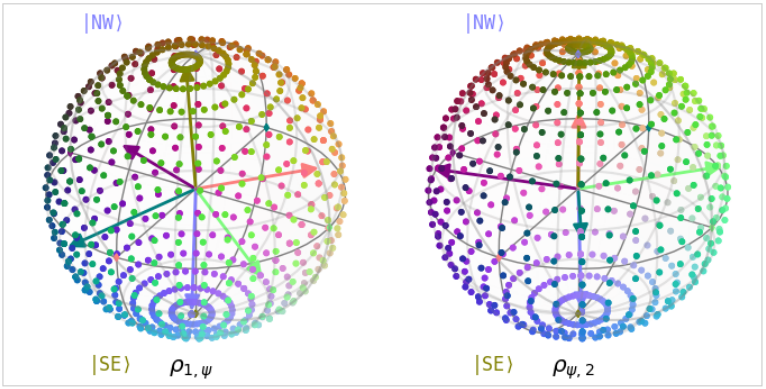


Figure 7.15: Demonstration that axiom (7.3) does not hold for this model. We display the very different results obtained by applying the (a) right and (b) left hand diagram fragments to a $|\texttt{South, South}\rangle$ initial state. The resulting states are respectively approximately a classical ensemble $\frac{1}{2}(|00\rangle\langle 00| + |11\rangle\langle 11|)$, and the pure, separable state $|ii\rangle\langle ii|$. This reinforces the finding that `follows` involves a projection, and highlights the inherent difficulty faced by the four directional model in encoding two bits of information into a single qubit.



(a) *goes in the same direction*



(b) *does not go in the same direction*

Figure 7.16: Visualising the (a) positive and (b) negative assertion states for the four directional model. The main directions distinguished by the model are labelled and coloured according to the state identified. These are all maximally entangled pure states. This model exhibits similar behaviour to the two-directional one, with similar, though less prominent, skews (recall that $|\texttt{NW}\rangle \approx |0\rangle$ for this model), such that `same dir` favours $|\texttt{SW, SW}\rangle$ and `not same dir` favours $|\texttt{NE, SW}\rangle$.

Questions

The assertion states are again very similar to the two directions model. We can express them in terms of the $\{|\texttt{NW}\rangle, |\texttt{SE}\rangle\}$ basis derived from the two-qubit `follows` projections (see Figure 7.16):

$$|\texttt{same dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{NW}\rangle \otimes |\texttt{NW}\rangle - |\texttt{SE}\rangle \otimes |\texttt{SE}\rangle)$$

$$|\texttt{opp dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{NW}\rangle \otimes |\texttt{SE}\rangle - e^{-i\pi/8} |\texttt{SE}\rangle \otimes |\texttt{NW}\rangle)$$

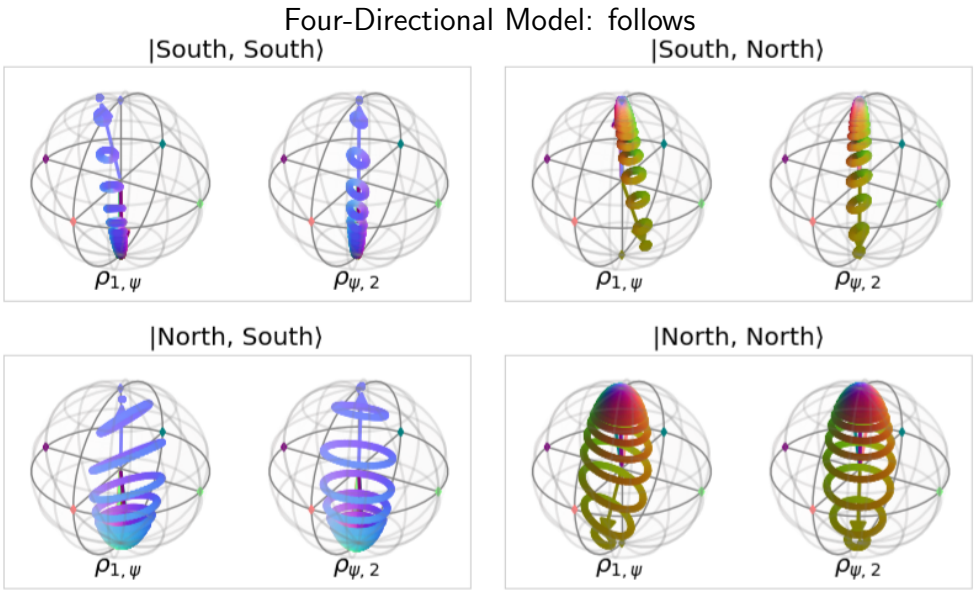
Once again, this shows the underlying two-directional nature of the model, in that it tests for correlation or anti-correlation along the $\{|\texttt{NW}\rangle, |\texttt{SE}\rangle\}$ basis.

7.1.4 Discussion

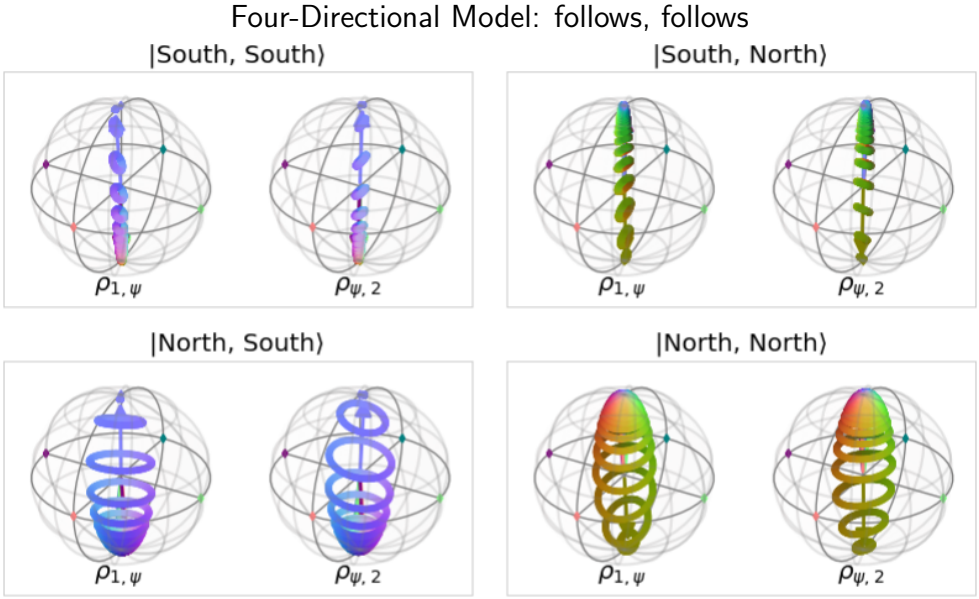
The analysis above helps us to identify and explain biases that the model exhibits. The bias can be seen to arise from a bias in the dataset - there are very few instances where the final directions faced by the model are at right angles to each other, so the model can afford to ignore these instances at relatively little cost. We can hence identify the instances that are difficult for the model as those that depend on the correct resolution of a quarter turn at some point in the text. As these are a relative minority across the dataset, the model appears to perform better than it should. This also means that the denser datasets will typically be easier for the model to solve (as exhibited in Figure 6.4), as they contain fewer quarter turns. For such datasets, the primary way for the chosen actors to be facing directions at right angles to each other at the end of the story is if they are tensor separate (as described in Section 3.2.2).

It is in fact expected that the model should struggle to perform well, as it is trying to encode 2 bits of information (which of the four directions each actor is currently facing) into a single qubit. The equivalent deterministic model for the four directional dataset (described in Figure 7.3) requires two qubits.

The fact that we were able to visualise the model in a modular way, and thus understand *why* it appeared to work, allowed us to identify a bias in our dataset. This was possible in part due to the relative simplicity of the model, as well as its compositional nature, which allowed us to consider only one and two qubit components of the model in isolation, and from this infer its action on arbitrary texts. In the two-directions case, this broadly supported the empirical evidence that the model had correctly learnt to generalise, whilst also exposing a failure case to be wary of. In contrast, for four directions we were able to identify that the model had *not* learnt to generalise in the way we had hoped, despite the otherwise promising empirical evidence.



(a)



(b)

Figure 7.17: Further spheres to visualise the effect of the two qubit `follows` gate of the four-directional model, for combinations covering an approximate two-qubit basis given by the `North` and `South` states. (a) shows a single application, while (b) shows two subsequent applications of the `follows` gate. The label $|x, y\rangle$ identifies the state $U(\theta_{\texttt{follows}})\,|x, y\rangle$. The expected joint final state is $|y, y\rangle$. The dependence on the state of the second qubit can be seen. Refer back to Figure 7.14.

7.2 bAbI 6

For the bAbI 6 tasks, we focus on an axiomatic approach, rather than investigating the quantum internals like for the *following* models, as a demonstration of the kind of approach that would also be possible for larger-scale models.

For this we make use of the notion of overlap for quantum states, and the map-state duality (depicted in Figure 7.18 and Figure 7.19 respectively) in order to directly compare learnt boxes and diagram fragments with one another.

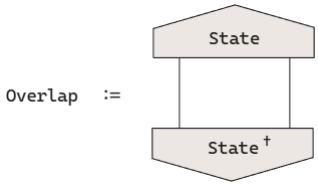


Figure 7.18: Computing the overlap between states. Requires a notion of adjoints, or compact closure. The adjoint of a state is an effect, labelled with a dagger (†).

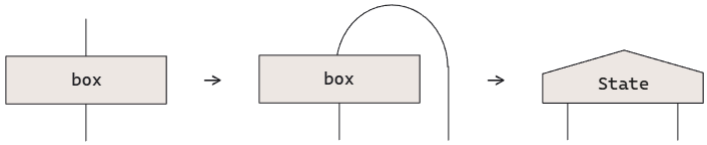


Figure 7.19: Converting a box into a state. This requires a rigid category.

We consider the two and three noun diagram fragments shown in Figure 7.25, Figure 7.28, Figure 7.29 and Figure 7.30. For a simpler depiction, we simplify the movement frame into a single box as per Figure 7.20. Similarly, we condense the object interaction verbs into a single box `grabs`.

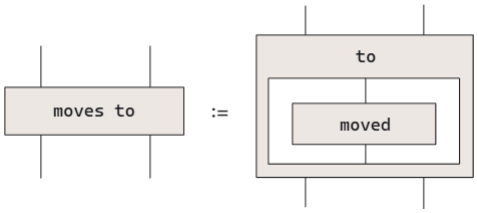


Figure 7.20: Simplification of the depiction of *moved to*, where `moved` stands for any of the intransitive movement verbs.

We also note that due to the task construction, we are interested in checking whether diagram fragments are equivalent *relative to the question*, as what goes

on in the rest of the space is effectively irrelevant. We compute assertion-relative overlaps by considering states in the format described in Figure 7.21, for each of the assertions relevant to answering the question.[1] An assertion state is appended to the diagram fragments as per the usual rules, with any extra wires discarded. We can then compute the overlaps between the resulting diagram fragment effects, as per Figure 7.18.

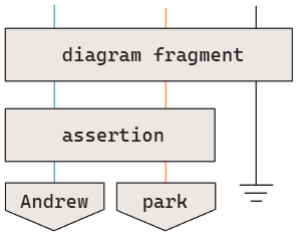


Figure 7.21: An assertion-relative diagram fragment. For this dataset the `assertion` box can be either *is in* or *is not in*.

7.2.1 Productivity: quantum model

Nouns

Visualising the distances between the nouns in Figure 7.22, we find the the model has learnt to group the people and places together, though remains largely agnostic when it comes to the objects. This is somewhat to be expected, as the task structure provides very little information to constrain the embedding of the objects, since they are not relevant to the task solution.

Movement verbs

We first consider the inner movement verbs that apply to only a single actor, then the composite *moves to*. We find that the intransitive movement verbs can be classed into two equivalent groups - this comes some way to realising the synonyms present in the actual task (in which all movement verbs are equivalent). Inspecting the symmetries, we also note that one group - the larger one - acts as approximately the identity. The overlaps are visualised in Figure 7.23.

[1] Recall from Section 2.3 that assertions are statements derived from the question, for each possible answer.

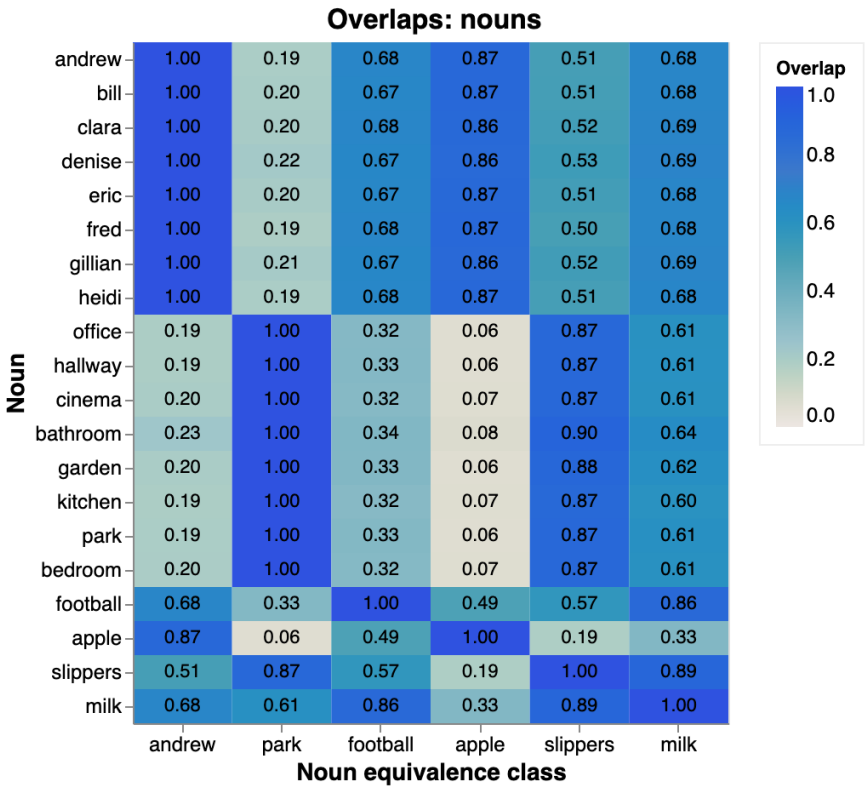


Figure 7.22: Visualising the relative overlaps between nouns. On the x-axis, we group the nouns by equivalence class, while the y-axis displays each of the nouns considered.

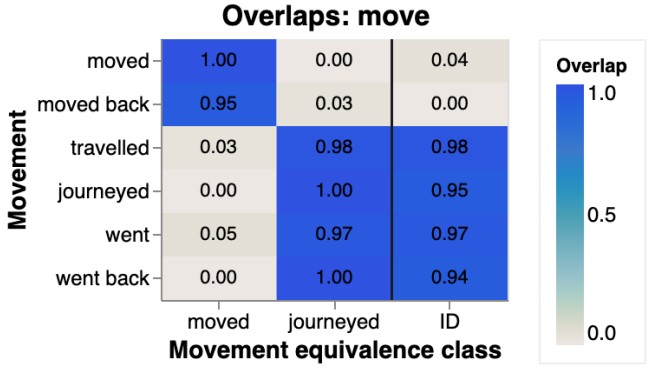


Figure 7.23: Visualising the relative overlaps between the intransitive *move*-type verbs. On the x-axis, we group the nouns by equivalence class, and include the identity for reference. The two classes are approximately orthogonal.

Object interaction verbs

Like the object noun states, these verbs are also less constrained by the dataset, as they do not contribute to the final answer but rather act as confounding information. Plotting their overlaps with respect to each other directly, we find that they are almost all small. A more instructive comparison is then to take their overlaps relative to the assertion states, plotted in Figure 7.24. We find that relative to these states, all *grab* type words are very similar, while there is more variation for the *drop*-type words. This suggests that the model has come some way to identifying these verbs as synonyms, despite the relatively low contribution these verbs have to the overall task.

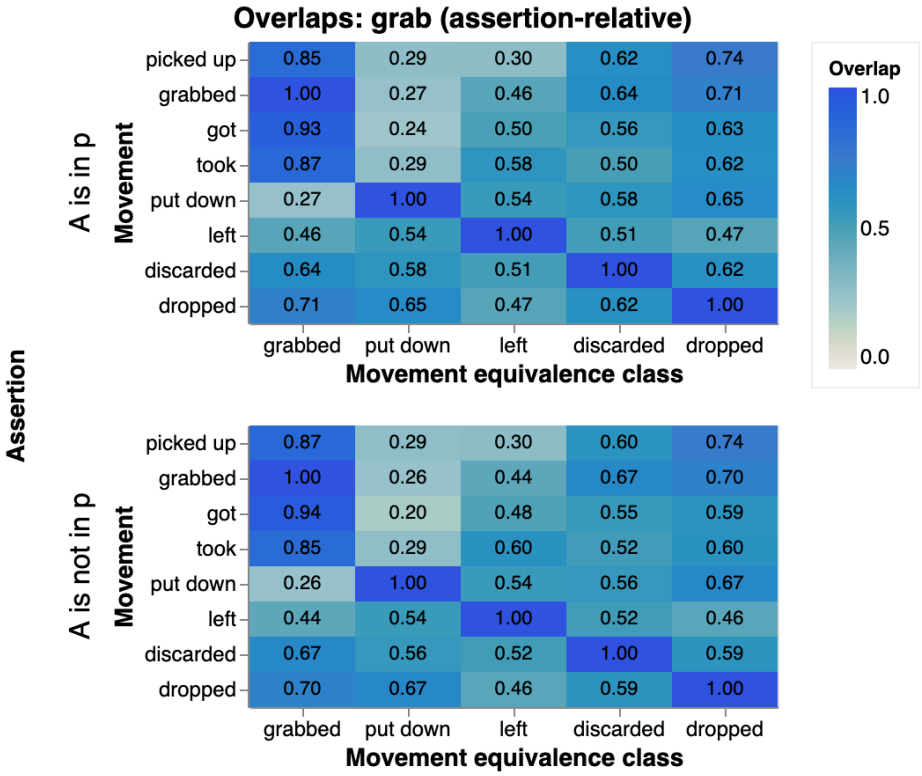


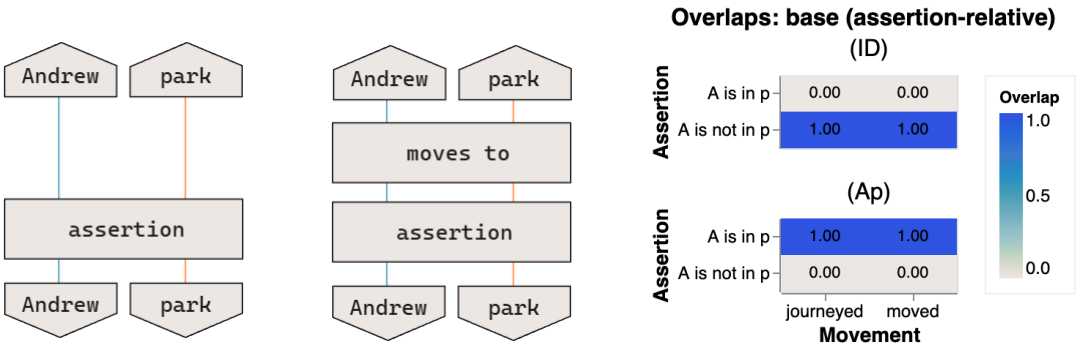
Figure 7.24: Visualising the assertion-relative overlaps between the transitive *grab*- and *drop*-type verbs. On the x-axis, we group the nouns by equivalence class.

Questions

As a binary task, there are again two possible question assertions. Since we find that there is approximately one person state, and one location state, we can express the assertions directly as states. We find these to be approximately orthogonal, with an overlap of 0.00053.

Comparing diagram fragments

We consider a series of diagram fragments, and their overlaps relative to the question assertions. Note that checking these overlaps relative to the question states, is not the same as checking the overlaps between the diagram fragments and the questions, as we would compute when actually solving the task. We consider a representative of each of the movement verb equivalence classes, though not mixed applications.



(a) (ID) (b) (Ap) (c)

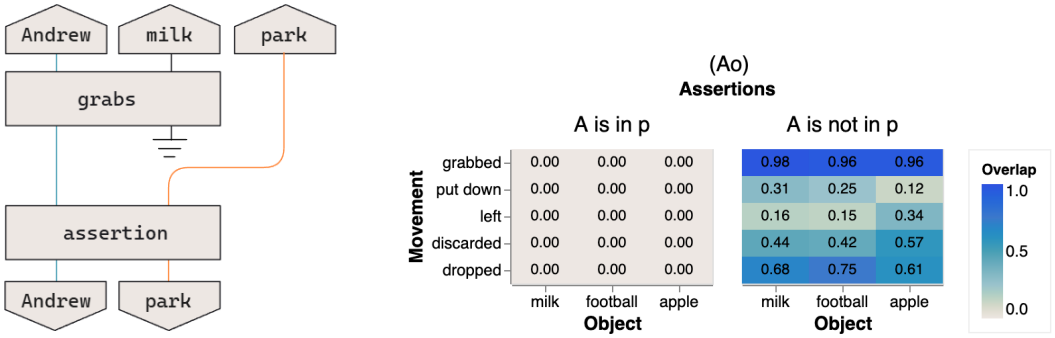
Figure 7.25: (a), (b) The diagrams evaluated to compute the base cases, and their shorthand identifiers. The wires corresponding to *Andrew* and *park* are coloured to help distinguish them. (c) Visualising the magnitude of the overlaps between the states, relative to each question state.

Base cases First, we consider the base cases, simulating the simplest texts. The diagram fragments and associated results are given in Figure 7.25 and Figure 7.26b.

(Ap) Base case: '*Andrew is in the park.*' The model is able to correctly identify the positive base case. Indeed, it seems that the question states are directly aligned with this state, as they achieve the maximum/minumum possible scores.

(ID) Base case: '*Andrew is not in the park.*' I As with the positive case, the model correctly identifies this instance, with the states again lining up approximately perfectly. Interestingly, such a situation never occurs in the dataset, as it would require a character to be introduced without doing anything, so this is a new rule that the model has successfully extrapolated.

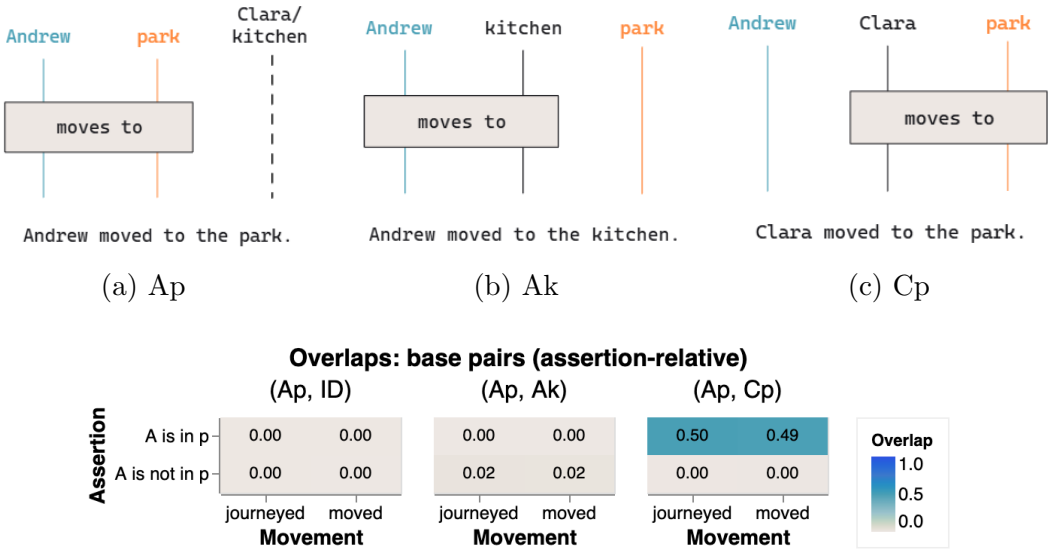
(Ao) Base case: '*Andrew is not in the park.*' II One situation that does occur in the dataset is for a person to be introduced via an object interaction and nothing else. We hence check how the model handles this situation as a secondary base case for the negative assertion, and find that the model captures this well for the *grab*-type verbs, but with a lower margin for the *drop*-type verbs. Though



(a) (Ao) (b)

Figure 7.26: (a) The base diagram fragment (Ao). (b) Visualising the magnitude of the overlaps between the diagram fragment (Ao) and the assertion states, for each choice of action and object. The values have been renormalised such that the maximum overlap remains 1.

the object embeddings were each distinct, we also find that this difference is no longer evident when factoring through the question states, suggesting the model has learnt to consider them equivalent.



(a) Ap (b) Ak (c) Cp

(d)

Figure 7.27: The selection of single-action diagram fragments considered and their overlaps. (a)-(c) The diagram fragments and their shorthand identifiers. The *Andrew* and *park* wires are coloured for emphasis. (d) Visualising the magnitude of the overlaps between the assertion-relative diagram fragments. For mixed states, the overlaps have been renormalised so that the maximum remains 1.

Singletons Next, we compare fragments with at most one verb, shown in Figure 7.27.

(Ap $\neq$ ID) Moving matters. Here, we check whether the act of moving has consequences. We see that the overlap between the states is very small for both assertion states, which suggests that we cannot simply delete a direct movement without affecting the final answer.

(Ap $\neq$ Ak) Distinguishable locations. This axiom captures the idea that it matters which place Andrew visits. In this case, all overlaps are close to zero, suggesting that the model is indeed sensitive to which location the target person visits.

(Ap $\neq$ Cp) Distinguishable people. For this pair, we check whether which person visits the target location has an impact. The results in this case have a similar structure for the different movement verbs, but the overlaps differ depending on the assertion state. In both cases, we find the overlap is zero relative to the negative assertion, and large relative to the positive one. This suggests, that changing *who* visited a location will have a much larger impact on the overlap between the story with the negative assertion than the positive one.

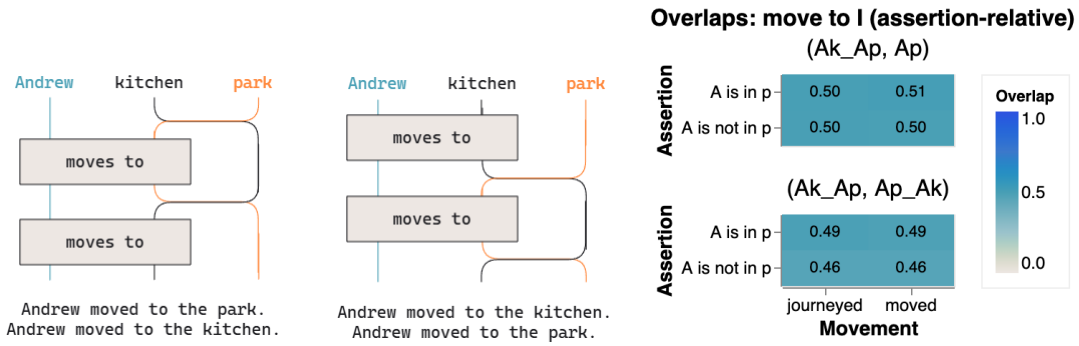
While intuitively we might like both overlaps to be close to zero, this does not necessarily mean that the model is *insensitive* to which person visits a location. This is possible since we only care about the *difference* between the assertion overlaps when computing the model's answers, such that a sufficient change in only one of the overlaps may be enough to change the model's final answer, provided the fixed overlap was not too extreme to start with. Though we did not explicitly check this here, comparing the fragments relative to *different* assertion states would be a way to measure whether the changes identified are in the correct direction.

Extra location Thirdly, we consider cases where the target person moves to a different location, as per Figure 7.28.

(Ap-Ak $\neq$ Ak-Ap) Visit ordering. We expect this pair to have low overlap for a satisfied axiom, reflecting that the order in which places are visited by a person matters. Instead, we find a fairly high overlap between the states, suggesting that the model is perhaps not distinguishing this as as well as it should.

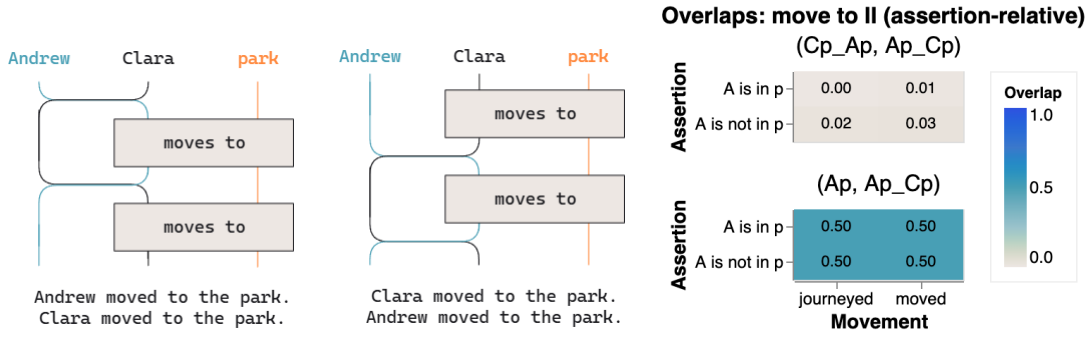
(Ak-Ap = Ap) Forgetfulness (other locations). For the purposes of this task, we do not need to remember where a person has been, only where they are now. This axiom captures the idea that we can forget where they were previously to entering the target location. The overlaps are all roughly 0.5 here, suggesting that

the model does not manage to forget locations very well. This result is somewhat unsurprising given the unitary nature of the model.



(a) Ap-Ak (b) Ak-Ap (c)

Figure 7.28: The selection of three noun diagram fragments considered for movement verbs involving a second location, and the paired assertion-relative overlaps. (a), (b) The diagram fragments and their shorthand identifiers. The wires corresponding to *Andrew* and *park* are coloured to help distinguish them. (c) Visualising the magnitude of the overlaps between the states in, relative to each assertion state. For mixed states, the overlaps have been renormalised so that the maximum remains 1.



(a) Ap-Cp (b) Cp-Ap (c)

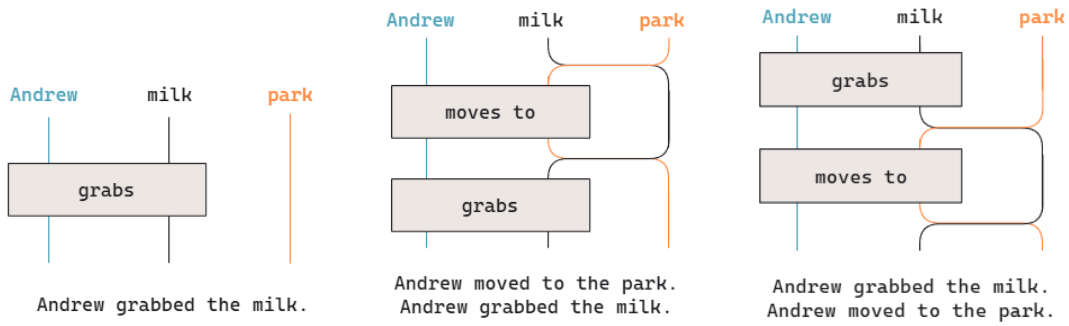
Figure 7.29: Diagram fragments and overlaps for movement combinations involving a second person. (a), (b) The selection of diagram fragments considered, and their shorthand identifiers. The wires corresponding to *Andrew* and *park* are coloured for emphasis. (c) Visualising the assertion-relative overlaps between the states. The overlaps have been renormalised so that the maximum remains 1.

Extra person Next, we consider cases where another person enters the target location, as depicted in Figure 7.29.

(Cp-Ap = Ap-Cp) Arrival order. Here we expect a high overlap, as the order in which people arrive at the target location should have no influence on the target

person's presence in that location. The model displays low overlaps with both assertions, however, suggesting that other people visiting the target location are likely to confuse it.

(Ap-Cp = Ap) Forgetfulness (other people). Next, we consider whether the model finds the presence of other people in a location to be important. As with the previous forgetfulness axiom, the values are around 0.5, suggesting that the model struggles to ignore the presence of other people in the target location completely. Combined with the behaviour of the (Cp-Ap, Ap-Cp) pair, we can conclude that the model may be quite sensitive to the presence or absence of other people in the target location.



(a) Ao (b) Ap-Ao (c) Ao-Ap

Figure 7.30: The selection of three noun diagram fragments considered for movement verbs and object interactions, with their shorthand identifiers. The wires corresponding to *Andrew* and *park* are coloured to help distinguish them.

Extra object Finally, we consider adding an object interaction. We display the diagram fragments in Figure 7.30, and the results in Figure 7.31.

(Ao-Ap = Ap) Forgetfulness (object interactions). The third kind of interaction to ignore are those with objects. Figure 7.31 visualises the overlaps for the object interactions. We find that the model behaves similarly across the different *move*-type verbs, but quite differently over the the *grab*- and *drop*-type verbs. For *grab*-type verbs, the model appears to successfully ignore the object interaction, while the *drop*-type verbs have lower overlaps, especially *put down*. This difference may be due to the sparsity of such instances in the data.

(Ap-Ao = Ao-Ap) Teleporting objects. For the purposes of this task, it does not matter where a person is when they interact with an object (indeed the object interactions do not matter at all). The model appears to have largely captured this invariance for the *grab*-type words, but again struggles for the *drop*-type verbs, especially *put down*.

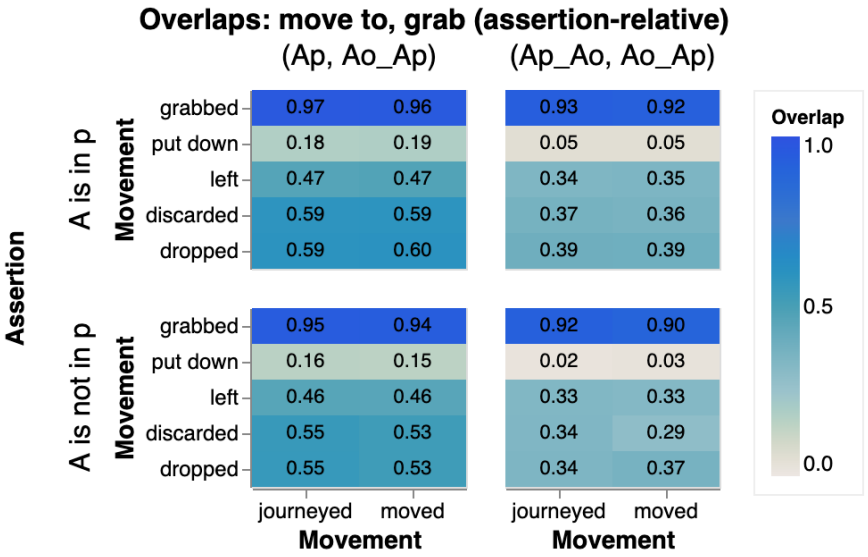


Figure 7.31: Visualising the magnitude of the assertion-relative overlaps between diagram fragments involving object interactions, from Figure 7.30. The values have been renormalised such that the maximum overlap is 1.

To summarise these investigations, we find that the model does not distinguish between *when* a person has entered a room very well, nor *who* has entered the room. This suggests that the model learns to answer a question closer to '*Is somebody in the park?*' rather than the expected '*Is Andrew in the park?*'. Indeed, we can use this metric to split the model's accuracy, as shown in Figure 7.32. When there is another person present in the target location, the model tends to answer incorrectly, most often by incorrectly answering *yes*.

This can then explain why the model struggled with productive generalisation, as the longer contexts tend to have more sentences involving the target person and location.

7.2.2 Discussion

In this section, we have demonstrated how to interpret a model, without relying on understanding or visualising the action of each component box individually. This approach, although allowing for a less deep understanding than that obtained for the *following* models in Section 7.1, is much more scalable to situations where each box applies to more than just two qubits.

Though the model we study did not do a great job of capturing the desirable axioms, we attribute this largely to limitations of the dataset and the relatively small scale at which the experiments ran. Despite this, we were nevertheless

able to explain how the model performed as it did, and identify the kind of situations that made the model fail. This was possible due to the inherent syntactic compositionality of the DisCoCirc model, which in turn allowed us to leverage the compositional understandability to construct relevant diagram fragments with which to test the model.

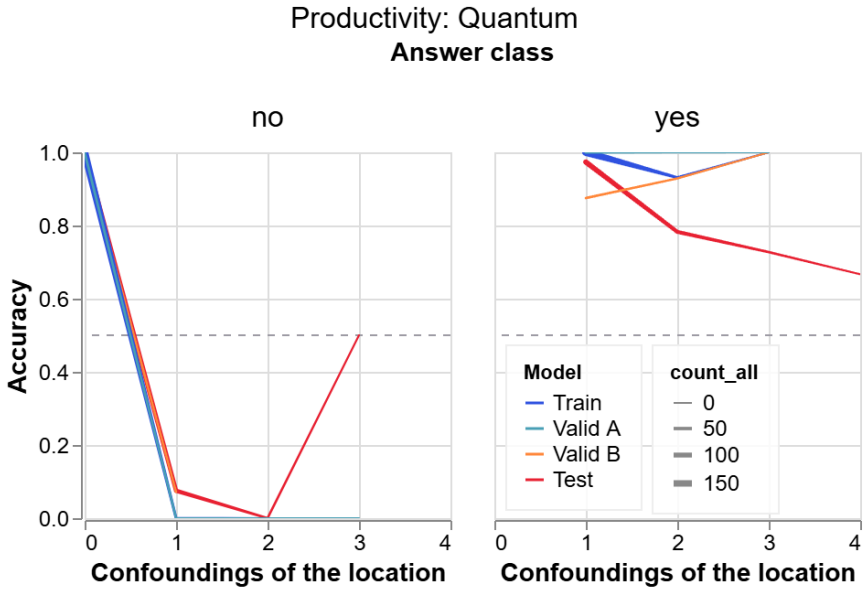


Figure 7.32: Visualising the model's accuracy per number of sentences in the context that include the target location. For instances where the expected answer is *yes*, the supporting sentence itself is counted as a confounding, which is why there are no 0 confoundings in that case. The line thickness represents the number of datapoints present.



CHAPTER 8

Conclusions

At the start of this thesis, we set out to explore DisCoCirc models for natural language processing, with a particular focus on their solution to two problems: the model's own 'understanding' of a text, and our own understanding of the model. Compositionality presents us with a potential solution to these two concerns. A model that compositionally generalises is able to correctly answer unseen claims about a given task, and in particular has captured the relevant compositional rules and relationships that underpin the task: in this sense, the model has achieved a notion of understanding for the given compositional task. Meanwhile, compositional understandability is what can allow us to inspect the models and derive an understanding of how the model arrives at an answer for ourselves. This work presents a first proof of concept exploration of DisCoCirc models, demonstrating both compositional generalisation and understandability. Here we review the work covered, and consider future directions of study.

Through the lens of two tasks (*following* and bAbI 6), we exhibited the capacities of DisCoCirc models. Firstly, we constructed a first proof-of-concept implementation of the quantum DisCoCirc model on the *following* datasets. The models demonstrated robust compositional generalisation (Section 6.1), achieving a productivity score of 1.0 and 0.76 for the two- and four-directional models respectively.

Next, we also investigated how well the DisCoCirc models could learn to accommodate three further aspects of compositionality: systematicity, substitutivity, and overgeneralisation identified in Section 1.2. We implemented both a quantum model, extending the results from the *following* dataset, as well as two neural architectures for comparison. Both versions of the model performed similarly

well on the bAbI productivity and substitutivity tasks (Section 6.2), achieving an accuracy of about $0.9 - 0.97$ on the *Train* set, and $0.7 - 0.8$ on the *Test* sets, leading to compositionality scores of about 0.52 for productivity and 0.65 for substitutivity. The models differed more significantly in performance over the systematicity task, with the neural architectures reaching a compositionality score of at most 0.5 while both quantum models were above 0.6. The two architectures also exhibited different trends on the overgeneralisation task, with the quantum model appearing to overgeneralise for low corruption levels, while the neural model overfit the data more significantly. We also noticed that the classical model hyperparameters were much more sensitive to variations in the task, and required extensive tuning.

Though we notice a decay in model performance when moving from the *following* task to the bAbI 6 tasks, we note that the bAbI tasks contained significantly fewer simplifying assumptions compared to the *following* task, such that the learning process was much more difficult in the bAbI case. Despite this, the models still exhibit promising signs of compositional generalisation, to be further refined in future work.

Beyond compositional generalisation, we also investigated a selection of trained models in detail, leveraging the inherent compositional understandability of the models in Chapter 7. This allowed us to confirm that when a model performed well, it was for the correct reasons (as with the two-directional *following* model in Section 7.1.2), while for models that had lower accuracy, we were able to identify how each model took advantage of biases in the dataset in order to simplify the task (for the four-directional *following* model in Section 7.1.3, and the quantum productivity model in Section 7.2).

8.1 Discussion

The significance of quantum

This work explores a selection of models following the DisCoCirc framework, one of which is implemented using a quantum architecture. Choosing such an approach has two main motivations: the first is historical - coming from the background of DisCoCat, a setting which required a compact-closed category (cups and caps), a quantum implementation becomes a natural choice as this setting (and in particular the $\otimes$ tensor product) is also compact closed. While moving to the DisCoCirc setting removes the majority of the cups that occur

within the text, question answering requires a notion of overlap, which is suitably captured by the adjoints present in the quantum setting.

The second motivation concerns the expressivity of the circuits - $\otimes$ based architectures allow for a greater variety of correlations between the entities involved, which in turn allows for a more expressive model. In Section 1.3 we considered how certain apparent counterexamples to compositionality could be accomodated by allowing the individual word meanings to be more expressive (for example, by modelling ambiguity using density matrices). The quantum setting may thus allow for more efficient route to implementing a model of the desired expressivity.

On the other hand, current quantum computers do not quite have the capacity to run a full scale model on realistic texts, so the utility of training such models remains somewhat restricted. For this reason, exploring classically tractable alternative architectures is important in the short term, in order to progress towards larger-scale data, and investigate how the model copes at scales beyond the toy settings investigated so far.

Alternative approaches

While we focused on the DisCoCirc framework, there are many other candidate models when it comes to solving NLP tasks, many of which are more mature frameworks. Most notably, transformer-based architectures are the current state of the art, though as established in Chapter 0, these models tend to struggle with compositional reasoning (Mirzadeh et al., 2024; Dziri et al., 2023).

Indeed, in Duneau et al. (2024), both a pre-trained LLM (GPT-4), and in-task trained transformer and LSTM models are evaluated on the *following* task reported on in this work. These models all under-perform compared to the DisCoCirc model, particularly when it comes to compositional generalisation. While this is in part due to some architecture constraints that allowed for greater simplification of the task in the case of the DisCoCirc model, and thus precludes a direct comparison between the models, this difference may nevertheless suggest that there is room for an alternative architecture when it comes to compositional tasks in particular.

8.2 Future work

At various stages of completing this thesis, many potential avenues for future research were encountered. We summarise them below:

Question formats

In this work, we considered DisCoCirc-style questions with explicit assertions. However, these require post-selections, and can become inefficient if there are many possible answers to a question. Additionally, these require a notion of adjoints or compact closure, which the standard category of neural networks does not include natively. DisCoCat-style questions (as per Section 2.3) provide an alternative paradigm, where the final answer can be read off from the output bitstring directly, without relying on the notion of an overlap. Along these lines, quantum dense-coding-friendly questions (in which multiple questions are used to build up a single answer bitstring) could also be explored to help leverage more of the quantum-specific properties.

Architectures with $\otimes$

Throughout this work, we explored a quantum model as a representative model for taking the monoidal product to be the tensor product $\otimes$, and both Linear and non-linear neural network models as representatives for the Cartesian product $\oplus$. There are, however many more options to consider.

For $\otimes$, quantum circuits were chosen as a trade-off between expressivity and ease of evaluation (on the assumption that a quantum computer is available at test time), however we heavily restricted the use of ancillas when implementing the models. This in turn restricted the expressibility of the models. A first direction to explore would then be the addition of more ancillas to gates, or perhaps allowing each noun to contain an extra qubit that is ultimately discarded.

Moving further from the quantum circuits studied here, we can also explore classes of models falling within more desirable complexity classes. In the easier-to-evaluate direction, a first choice might be to explore Clifford circuits. While these are easy to simulate, we expect training to be more difficult as optimisation must occur over a discrete training space.

A more promising approach in this direction is then free-fermion based circuits (Surace and Tagliacozzo, 2022) - a restricted class of quantum circuit that is both easy to simulate and trainable with continuous parameters. These circuits also admit relaxations that increase expressivity at the cost of simulability in a controlled manner, which can allow for a more nuanced trade-off to be reached. More in-depth exploration of the relative performance of such models might help to shed some light onto how much (if at all) the DisCo type models rely on

quantum-specific features, both to help with their performance at compositional generalisation, as well as for allowing compositional understandability.

Going in the other direction, we could also investigate quantum circuit classes that are known to be classically hard to simulate, such as instantaneous quantum polynomial (IQP) circuits - finding a task for which such models outperform classical models may be an avenue for investigating quantum advantage. Less optimistically, we might also consider training unconstrained tensor networks directly. While these may be harder to simulate, they are more expressive and would allow models to solve more complex tasks than when restricting to practical[1] quantum circuits.

Having generalised to generic tensor networks, we can also chose to restrict these in a different way, to retrieve a class that is easy to simulate, for example by considering classical probability distributions. See Glasser et al. (2019) for an introductory overview.

Architectures with $\oplus$

In this work we considered two neural network based architectures, however both were ultimately only considered on a very small scale. One option to explore would then be to scale up the components - exploring deeper and wider networks for each box to take better advantage of the relative efficiency for running the $\oplus$ models. Another line to explore are more sophisticated, or different activation functions. A notable example that was not studied here were the Sigmoid-type activations.

While the quantum hyperparameters were relatively robust across tasks, the neural models were much more sensitive: further exploration could be useful to either find more robust hyperparameters, or determine better heuristics for selecting them.

Datasets

One trend that has remained robust in the history of NLP is that more data leads to better performing models. In this proof-of-concept work, the tasks studied were all quite small, with dataset sizes on the order of 1000 datapoints. Scaling up to larger datasets would likely require different architecture choices to ensure

[1]We take this to mean quantum circuits with limited post-selections and ancilla wires, whose evaluation does not require an exponential number of shots as the size of the circuit increases.

that training remains practical, but would allow the exploration of more realistic datasets with larger vocabularies. A first step could be to address multiple bAbI tasks at once, encouraging the model to learn more sophisticated embeddings that can be used with a selection of different questions, as well as considering multiple compositionality aspects at once.

Looking a little further, we might also consider more realistic reading comprehension datasets, such as the RACE dataset presented by Lai et al. (2017). This dataset compiles English reading comprehension questions aimed at Chinese middle-school students, with answers in a multiple choice format, making it well suited for DisCoCirc type models.

Training paradigms

When considering training paradigms to experiment with, we identified meta-learning as a promising next step, beyond the curriculum learning explored here. The results in this work suggest that while models can perform very well if a suitably semantically compositional model can be found, finding such a model can be difficult, especially when the training does not reliably produce a large set of good models to choose from. Further work is then required on the methodology side in order to help the training process produce more good models, as well as at the validation stage in order to reliably select these models. The meta-learning paradigm is framed specifically for training on compositional tasks in Conklin et al. (2021).

Rewrites and axioms

Here, we explored two somewhat extreme cases of rewrites. In the following task, a great number of such rewrites were provided to the model during training, which made the optimisation much easier. The bAbI tasks, on the other hand, were explicitly testing how well the model could learn such rewrites by itself. Though the model certainly came some way to capturing these rewrites, it is clear from the comparative scores that this was not a trivial task. As part of exploring alternative and larger datasets, we may also consider how rewrites and axioms might be integrated directly into the training in order to give the model an effective head start.

8.2.1 Beyond

This work provides the first proof-of-concept for DisCoCirc architectures, paving the way for more sophisticated approaches to tackle more realistic tasks. So far we have considered relatively local directions for future exploration, however there remain some broader, more complicated questions to explore.

Quantifying the difference between $\otimes$ and $\oplus$ architectures

This likely requires both theoretical and experimental validation. In this work, we did not notice a significant trade-off in compositional behaviour or accuracy between the two, though intuitively we would expect $\otimes$-based architectures to perform better. Currently, it remains an open question to characterise this difference more specifically. Possibly, certain larger-scale tasks might benefit from the larger encoding space provided by $\otimes$-based models.

Generation

Secondly, in this work we have not addressed the parsing or generation aspects of the model, other than via the assumption that a parsing can be obtained, and that assertions can be generated from questions. Other work, most notably Wang-Mascianica (2023) explores both parsing and generation in more detail. In the context of the question answering tasks studied here, such methods might be used to automatically derive assertions from questions, or even to generate free-text responses to more open-ended questions.

DisCoCirc as text embeddings

In their original form, the DisCo- frameworks are inherently methods for computing text embeddings. Taking this seriously, we might investigate whether pre-trained DisCoCirc embeddings provide any benefit relative to other methods such as BERT (Devlin et al., 2018) and GloVe (Pennington et al., 2014b) when solving a downstream task. Though the embeddings themselves might be interpretable, this approach would lose any special interpretability benefits conferred by the model structure depending on how the embeddings are processed; whether this is a worthwhile tradeoff remains to be empirically determined.

Beyond natural language

Many other domains enjoy compositional structures. While the DisCo- frameworks were originally theorised as models of natural language, we might also consider alternative grammars that describe other structured domains where compositionality is desirable. Graphs are a natural generalisation of natural language parsing trees, and could hence admit natural DisCo- embeddings.

Bibliography

S. Alouneh, S. Abed, M. H. Al Shayeji, and R. Mesleh. A comprehensive study and analysis on SAT-solvers: advances, usages and achievements. Artificial Intelligence Review, 52(4):2575–2601, Dec. 2019. ISSN 1573-7462. doi: 10.1007/s10462-018-9628-0. URL https://doi.org/10.1007/s10462-018-9628-0.

J. B. Altepeter, E. R. Jeffrey, M. Medic, and P. Kumar. Multiple-qubit quantum state visualization. In 2009 Conference on Lasers and Electro-Optics and 2009 Conference on Quantum electronics and Laser Science Conference, pages 1–2, 2009. doi: 10.1364/IQEC.2009.IWC1.

K. Arkoudas. GPT-4 Can't Reason. 2023. doi: 10.48550/ARXIV.2308.03762. URL https://arxiv.org/abs/2308.03762. Publisher: arXiv Version Number: 2.

E. Bakshy, L. Dworkin, B. Karrer, K. Kashin, B. Letham, A. Murthy, and S. Singh. Ae: A domain-agnostic platform for adaptive experimentation. In NeurIPS Systems for ML Workshop, 2018. URL http://learningsys.org/nips18/assets/papers/87CameraReadySubmissionAE%20-%20NeurIPS%202018.pdf.

E. Balkir, D. Kartsaklis, and M. Sadrzadeh. Sentence Entailment in Compositional Distributional Semantics. Annals of Mathematics and Artificial Intelligence, 82 (4):189–218, Apr. 2018. ISSN 1012-2443, 1573-7470. doi: 10.1007/s10472-017-9570-x. URL http://arxiv.org/abs/1512.04419. arXiv: 1512.04419.

M. Balthasar Mansky, F. Wörle, J. Korbinian Stein, R. Müller, and C. Linnhoff-Popien. Adapting the DisCoCat-Model for Question Answering in the Chinese Language. In 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 01, pages 591–600, Sept. 2023. doi: 10.1109/QCE57702.2023.00073. URL https://ieeexplore.ieee.org/document/10313771/?arnumber=10313771.

D. Bankova, B. Coecke, M. Lewis, and D. Marsden. Graded Entailment for Compositional Distributional Semantics. arXiv:1601.04908 [quant-ph], Jan. 2016. URL http://arxiv.org/abs/1601.04908. arXiv: 1601.04908.

H. Barendregt and H. Geuvers. Chapter 18 - Proof-Assistants Using Dependent Type Systems. In A. Robinson and A. Voronkov, editors, Handbook

of Automated Reasoning, Handbook of Automated Reasoning, pages 1149–1238. North-Holland, Amsterdam, Jan. 2001. ISBN 978-0-444-50813-3. doi: 10.1016/B978-044450813-3/50020-5. URL https://www.sciencedirect.com/science/article/pii/B9780444508133500205.

A. R. Barron. Approximation and estimation bounds for artificial neural networks. Machine Learning, 14(1):115–133, Jan. 1994. ISSN 1573-0565. doi: 10.1007/BF00993164. URL https://doi.org/10.1007/BF00993164.

N. Block. Psychologism and Behaviorism. Philosophical Review, 90(1):5–43, 1981. doi: 10.2307/2184371. Publisher: Duke University Press.

M. J. Bremner, R. Jozsa, and D. J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 467 (2126):459–472, Feb. 2011. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.2010.0301. URL http://arxiv.org/abs/1005.1407. arXiv:1005.1407 [quant-ph].

M. J. Bremner, A. Montanaro, and D. J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. Physical Review Letters, 117(8):080501, Aug. 2016. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.117.080501. URL http://arxiv.org/abs/1504.07999. arXiv:1504.07999 [quant-ph].

S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of Artificial General Intelligence: Early experiments with GPT-4, Apr. 2023. URL http://arxiv.org/abs/2303.12712. arXiv:2303.12712 [cs].

N. Chomsky. Aspects of the theory of syntax. 1965.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika, 26(4):404–413, 1934. ISSN 00063444. URL http://www.jstor.org/stable/2331986.

B. Coecke. The Mathematics of Text Structure. arXiv:1904.03478 [quant-ph], Feb. 2020. URL http://arxiv.org/abs/1904.03478. arXiv: 1904.03478.

B. Coecke and A. Kissinger. Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning. Cambridge University Press, Cambridge, 2017. ISBN 978-1-316-21931-7. doi: 10.1017/9781316219317. URL http://ebooks.cambridge.org/ref/id/CBO9781316219317.

B. Coecke and V. Wang. Grammar equations. In Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace), pages 32–41, Groningen, The Netherlands, June 2021. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2021.semspace-1.4.

B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical Foundations for a Compositional Distributional Model of Meaning. arXiv:1003.4394 [cs, math], Mar. 2010. URL http://arxiv.org/abs/1003.4394. arXiv: 1003.4394.

B. Coecke, G. d. Felice, D. Marsden, and A. Toumi. Towards Compositional Distributional Discourse Analysis. Electronic Proceedings in Theoretical Computer Science, 283:1–12, Nov. 2018. ISSN 2075-2180. doi: 10.4204/EPTCS.283.1. URL http://arxiv.org/abs/1811.03277. arXiv:1811.03277 [cs].

H. Conklin, B. Wang, K. Smith, and I. Titov. Meta-Learning to Compositionally Generalize. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3322–3335, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.258. URL https://aclanthology.org/2021.acl-long.258.

A. D. Correia, M. Moortgat, and H. T. C. Stoof. Grover's Algorithm for Question Answering, Jan. 2022. URL http://arxiv.org/abs/2106.05299. arXiv:2106.05299 [quant-ph].

G. D. l. Cuevas, A. Klingler, M. Lewis, and T. Netzer. Cats climb entails mammals move: preserving hyponymy in compositional distributional semantics. arXiv:2005.14134 [cs, math], May 2020. URL http://arxiv.org/abs/2005.14134. arXiv: 2005.14134.

E. Davis and S. Aaronson. Testing GPT-4 with Wolfram Alpha and Code Interpreter plug-ins on math and science problems, Sept. 2024. URL http://arxiv.org/abs/2308.05713. arXiv:2308.05713 [cs].

G. de Felice, K. Meichanetzidis, and A. Toumi. Functorial Question Answering. arXiv:1905.07408 [cs, math], Sept. 2020. doi: 10.4204/EPTCS.323.6. URL http://arxiv.org/abs/1905.07408. arXiv: 1905.07408 version: 3.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

T. Duneau. Parsing conjunctions in DisCoCirc. In Proceedings of the 2021 Workshop on Semantic Spaces at the Intersection of NLP, Physics, and Cognitive Science (SemSpace), pages 66–75, Groningen, The Netherlands, June 2021a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2021.semspace-1.7.

T. Duneau. Solving Logical Puzzles in DisCoCirc. Journal of Cognitive Science, 22(3):355 – 389, 2021b. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85125423576&partnerID=40&md5=4c6db945056372334795ff799ea0f56f. Type: Article.

T. Duneau. Towards a Comparative Framework for Compositional AI Models, June 2025. URL http://arxiv.org/abs/2507.02940. arXiv:2507.02940 [cs].

T. Duneau, S. Bruhn, G. Matos, T. Laakkonen, K. Saiti, A. Pearson, K. Meichanetzidis, and B. Coecke. Scalable and interpretable quantum natural language processing: an implementation on trapped ions, Sept. 2024. URL http://arxiv.org/abs/2409.08777. arXiv:2409.08777.

N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, P. West, C. Bhagavatula, R. L. Bras, J. D. Hwang, S. Sanyal, S. Welleck, X. Ren, A. Ettinger, Z. Harchaoui, and Y. Choi. Faith and Fate: Limits of Transformers on Compositionality, Oct. 2023. URL http://arxiv.org/abs/2305.18654. arXiv:2305.18654 [cs].

R. Eldan and O. Shamir. The Power of Depth for Feedforward Neural Networks, May 2016. URL http://arxiv.org/abs/1512.03965. arXiv:1512.03965.

Facebook. Ax - adaptive experimentation platform. https://github.com/facebook/Ax.

M. Farkas, N. Miklin, and A. Tavakoli. Simple and general bounds on quantum random access codes. arXiv preprint arXiv:2312.14142, 2024.

C. Fellbaum, editor. WordNet: An Electronic Lexical Database. Language, Speech, and Communication. A Bradford Book, Cambridge, MA, USA, May 1998. ISBN 978-0-262-06197-1.

M. M. Ferdaus, M. Abdelguerfi, E. Ioup, K. N. Niles, K. Pathak, and S. Sloan. Towards Trustworthy AI: A Review of Ethical and Robust Large Language Models. 2024. doi: 10.48550/ARXIV.2407.13934. URL https://arxiv.org/abs/2407.13934. Publisher: arXiv Version Number: 1.

J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. Cognition, 28(1):3–71, Mar. 1988. ISSN 0010-0277. doi: 10.1016/0010-0277(88)90031-5. URL https://www.sciencedirect.com/science/article/pii/0010027788900315.

I. Glasser, R. Sweke, N. Pancotti, J. Eisert, and J. Cirac. Expressive power of tensor-network factorizations for probabilistic modeling, with applications from hidden Markov models to quantum machine learning. July 2019. URL https://www.semanticscholar.org/paper/0eb71a7262fe1726c5096380f3569878f2a8aaa7.

E. Grefenstette and M. Sadrzadeh. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1394–1404, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D11-1129.

H. P. Grice. Logic and Conversation. In D. Davidson, editor, The logic of grammar, pages 64–75. Dickenson Pub. Co., 1975.

P. Gärdenfors. Semantics Based on Conceptual Spaces. In M. Banerjee and A. Seth, editors, Logic and Its Applications, Lecture Notes in Computer Science, pages 1–11, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-18026-2. doi: 10.1007/978-3-642-18026-2_1.

R. F. Hadley. Systematicity in Connectionist Language Learning. Mind & Language, 9(3):247–272, 1994. ISSN 1468-0017. doi: 10.1111/j.1468-0017.1994.tb00225.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0017.1994.tb00225.x. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-0017.1994.tb00225.x.

J. Hefford and M. Wilson. A Profunctorial Semantics for Quantum Supermaps, May 2024. URL http://arxiv.org/abs/2402.02997. arXiv:2402.02997.

X. Huang, J. Zhang, D. Li, and P. Li. Knowledge Graph Embedding Based Question Answering. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM '19, pages 105–113, New York, NY, USA, Jan. 2019. Association for Computing Machinery. ISBN 978-1-4503-5940-5. doi: 10.1145/3289600.3290956. URL https://dl.acm.org/doi/10.1145/3289600.3290956.

D. Hupkes, V. Dankers, M. Mul, and E. Bruni. Compositionality decomposed: how do neural networks generalise?, Feb. 2020. URL http://arxiv.org/abs/1908.08351. arXiv:1908.08351 [cs, stat].

B. Jiang, Y. Xie, Z. Hao, X. Wang, T. Mallick, W. J. Su, C. J. Taylor, and D. Roth. A Peek into Token Bias: Large Language Models Are Not Yet Genuine Reasoners. 2024. doi: 10.48550/ARXIV.2406.11050. URL https://arxiv.org/abs/2406.11050. Publisher: arXiv Version Number: 2.

K. Johnson and Journal of Philosophy, Inc. On the Systematicity of Language and Thought:. Journal of Philosophy, 101(3):111–139, 2004. ISSN 0022-362X. doi: 10.5840/jphil2004101321. URL http://www.pdcnet.org/oom/service?url_ver=Z39.88-2004&rft_val_fmt=&rft.imuse_id=jphil_2004_0101_0003_0111_0139&svc_id=info:www.pdcnet.org/collection.

J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling Laws for Neural Language Models, Jan. 2020. URL http://arxiv.org/abs/2001.08361. arXiv:2001.08361 [cs].

D. Kartsaklis. Coordination in Categorical Compositional Distributional Semantics. Electronic Proceedings in Theoretical Computer Science, 221:29–38, Aug. 2016. ISSN 2075-2180. doi: 10.4204/EPTCS.221.4. URL http://arxiv.org/abs/1606.01515. arXiv: 1606.01515.

D. Kartsaklis and M. Sadrzadeh. Prior Disambiguation of Word Tensors for Constructing Sentence Vectors. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1590–1601, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1166.

D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, and B. Coecke. lambeq: An Efficient High-Level Python Library for Quantum NLP. Oct. 2021. URL https://arxiv.org/abs/2110.04236v1.

L. Karttunen. Discourse Referents. In International Conference on Computational Linguistics COLING 1969: Preprint No. 70, Sånga Säby, Sweden, Sept. 1969. URL https://aclanthology.org/C69-7001.

J. Kim. Explanatory Knowledge and Metaphysical Dependence. Philosophical Issues, 5:51–69, 1994. ISSN 1533-6077. doi: 10.2307/1522873. URL https://www.jstor.org/stable/1522873. Publisher: [Wiley, Dr. Enrique Villanueva, Ridgeview Publishing Company].

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

T. Laakkonen, K. Meichanetzidis, and B. Coecke. Quantum Algorithms for Compositional Text Processing. Electronic Proceedings in Theoretical Computer Science, 406:162–196, Aug. 2024. ISSN 2075-2180. doi: 10.4204/EPTCS.406.8. URL http://arxiv.org/abs/2408.06061. arXiv:2408.06061 [quant-ph].

G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. RACE: Large-scale ReAding Comprehension Dataset From Examinations. arXiv:1704.04683 [cs], Dec. 2017. URL http://arxiv.org/abs/1704.04683. arXiv: 1704.04683.

J. Lambek. Type Grammar Revisited. In A. Lecomte, F. Lamarche, and G. Perrier, editors, Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, pages 1–27, Berlin, Heidelberg, 1999. Springer. ISBN 978-3-540-48975-7. doi: 10.1007/3-540-48975-4_1.

J. Lambek. Pregroups and natural language processing. The Mathematical Intelligencer, 28(2):41–48, Mar. 2006. ISSN 0343-6993. doi: 10.1007/BF02987155. URL https://doi.org/10.1007/BF02987155.

J. Lambek. From Word to Sentence: A Computational Algebraic Approach to Grammar. Polimetrica s.a.s., 2008. ISBN 978-88-7699-117-2. Google-Books-ID: ZHgRaRaadJ4C.

M. Lewis. Modelling hyponymy for DisCoCat. 2019. URL https://www.cs.ox.ac.uk/ACT2019/preproceedings/Martha%20Lewis.pdf.

M. Lewis. Towards logical negation for compositional distributional semantics. arXiv:2005.04929 [cs, math], May 2020. URL http://arxiv.org/abs/2005.04929. arXiv: 2005.04929.

J. Liu, R. A. Shaikh, B. Rodatz, R. Yeung, and B. Coecke. A Pipeline For Discourse Circuits From CCG, Nov. 2023. URL http://arxiv.org/abs/2311.17892. arXiv:2311.17892 [cs].

R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke. QNLP in Practice: Running Compositional Models of Meaning on a Quantum Computer. arXiv:2102.12846 [quant-ph], Feb. 2021. URL http://arxiv.org/abs/2102.12846. arXiv: 2102.12846.

I. R. McKenzie, A. Lyzhov, M. Pieler, A. Parrish, A. Mueller, A. Prabhu, E. McLean, A. Kirtland, A. Ross, A. Liu, A. Gritsevskiy, D. Wurgaft, D. Kauffman, G. Recchia, J. Liu, J. Cavanagh, M. Weiss, S. Huang, T. F. Droid, T. Tseng, T. Korbak, X. Shen, Y. Zhang, Z. Zhou, N. Kim, S. R. Bowman, and E. Perez. Inverse Scaling: When Bigger Isn't Better, May 2024. URL http://arxiv.org/abs/2306.09479. arXiv:2306.09479 [cs].

K. Meichanetzidis, A. Toumi, G. de Felice, and B. Coecke. Grammar-Aware Question-Answering on Quantum Computers. arXiv:2012.03756 [quant-ph], Dec. 2020. URL http://arxiv.org/abs/2012.03756. arXiv: 2012.03756.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

I. Mirzadeh, K. Alizadeh, H. Shahrokhi, O. Tuzel, S. Bengio, and M. Farajtabar. GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models, Oct. 2024. URL http://arxiv.org/abs/2410.05229. arXiv:2410.05229 [cs].

R. Montague. Universal grammar. Theoria, 36:373–398, 1970.

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

OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh,

R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. d. A. B. Peres, M. Petrov, H. P. d. O. Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. J. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, and B. Zoph. GPT-4 Technical Report, Mar. 2024. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs].

B. Partee. Compositionality. Varieties of formal semantics, Jan. 1984a. URL https://www.academia.edu/3187795/Compositionality.

B. H. Partee. Nominal and Temporal Anaphora. Linguistics and Philosophy, 7 (3):243–286, 1984b. ISSN 0165-0157. URL https://www.jstor.org/stable/25001168. Publisher: Springer.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, Advances in Neural

Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

F. Pelletier. The Principle of Semantic Compositionality. Topoi, 13:11–24, Mar. 1994. doi: 10.1007/BF00763644.

J. Pennington, R. Socher, and C. Manning. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, 2014a. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL http://aclweb.org/anthology/D14-1162.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014b.

R. Piedeleu and F. Zanasi. An Introduction to String Diagrams for Computer Scientists, Nov. 2023. URL http://arxiv.org/abs/2305.08768. arXiv:2305.08768 [cs].

R. Piedeleu, D. Kartsaklis, B. Coecke, and M. Sadrzadeh. Open System Categorical Quantum Semantics in Natural Language Processing. arXiv:1502.00831 [cs, math], Feb. 2015. URL http://arxiv.org/abs/1502.00831. arXiv: 1502.00831.

Quantinuum. Quantinuum system model H1 product data sheet. https://assets.website-files.com/62b9d45fb3f64842a96c9686/654528cf9dd7824353bd9b56_QuantinuumH1ProductDataSheetv6.030Oct23.pdf, 2023. Accessed: 2024-02-27.

F. Recanati. Composition Ality, Flexibility, And Context Dependence. In W. Hinzen, E. Machery, and M. Werning, editors, The Oxford Handbook of Compositionality, page 0. Oxford University Press, Feb. 2012. ISBN 978-0-19-954107-2. doi: 10.1093/oxfordhb/9780199541072.013.0008. URL https://doi.org/10.1093/oxfordhb/9780199541072.013.0008.

D. Reutter and J. Vicary. High-level methods for homotopy construction in associative n-categories, Feb. 2019. URL http://arxiv.org/abs/1902.03831. arXiv:1902.03831 [math].

C. Roberts, A. Milsted, M. Ganahl, A. Zalcman, B. Fontaine, Y. Zou, J. Hidary, G. Vidal, and S. Leichenauer. Tensornetwork: A library for physics and machine learning. arXiv:1905.01330, 2019.

B. Rodatz, R. A. Shaikh, and L. Yeh. Conversational Negation using Worldly Context in Compositional Distributional Semantics. arXiv:2105.05748 [quant-ph], May 2021. URL http://arxiv.org/abs/2105.05748. arXiv: 2105.05748.

C. Rovelli. The Relational Interpretation of Quantum Physics, Sept. 2021. URL http://arxiv.org/abs/2109.09170. arXiv:2109.09170.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1 (5):206–215, May 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0048-x. URL https://www.nature.com/articles/s42256-019-0048-x. Publisher: Nature Publishing Group.

M. Sadrzadeh, S. Clark, and B. Coecke. The Frobenius anatomy of word meanings I: subject and object relative pronouns. Journal of Logic and Computation, 23(6): 1293–1317, Dec. 2013. ISSN 0955-792X, 1465-363X. doi: 10.1093/logcom/ext044. URL http://arxiv.org/abs/1404.5278. arXiv: 1404.5278.

M. Sadrzadeh, S. Clark, and B. Coecke. The Frobenius anatomy of word meanings II: possessive relative pronouns. Journal of Logic and Computation, 26(2): 785–815, Apr. 2016. ISSN 0955-792X, 1465-363X. doi: 10.1093/logcom/exu027. URL http://arxiv.org/abs/1406.4690. arXiv: 1406.4690.

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

R. A. Shaikh, L. Yeh, B. Rodatz, and B. Coecke. Composing Conversational Negation. arXiv:2107.06820 [cs, math], July 2021. URL http://arxiv.org/abs/2107.06820. arXiv: 2107.06820.

S. Sim, P. D. Johnson, and A. Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Advanced Quantum Technologies, 2(12), Oct. 2019. ISSN 2511-9044. doi: 10.1002/qute.201900070. URL http://dx.doi.org/10.1002/qute.201900070.

P. Smolensky. The Constituent Structure of Connectionist Mental States: A Reply to Fodor and Pylyshyn. In T. Horgan and J. Tienson, editors, Connectionism and the Philosophy of Mind, pages 281–308. Springer Netherlands, Dordrecht, 1991. ISBN 978-94-011-3524-5. doi: 10.1007/978-94-011-3524-5_13. URL https://doi.org/10.1007/978-94-011-3524-5_13.

J. Surace and L. Tagliacozzo. Fermionic Gaussian states: an introduction to numerical approaches. SciPost Physics Lecture Notes, page 54, May 2022. ISSN 2590-1990. doi: 10.21468/SciPostPhysLectNotes.54. URL http://arxiv.org/abs/2111.08343. arXiv:2111.08343 [quant-ph].

Z. G. Szabó. The case for compositionality. In W. Hinzen, E. Machery, and M. Werning, editors, The Oxford Handbook of Compositionality, page 0. Oxford University Press, Feb. 2012. ISBN 978-0-19-954107-2. doi: 10.1093/oxfordhb/9780199541072.013.0003. URL https://doi.org/10.1093/oxfordhb/9780199541072.013.0003.

R. Tamari, K. Richardson, A. Sar-Shalom, N. Kahlon, N. Liu, R. Tsarfaty, and D. Shahaf. Dyna-bAbI: unlocking bAbI's potential with dynamic synthetic benchmarking, Nov. 2021. URL http://arxiv.org/abs/2112.00086. arXiv:2112.00086.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and Efficient Foundation Language Models, Feb. 2023. URL http://arxiv.org/abs/2302.13971. arXiv:2302.13971 [cs].

S. Tull, R. Lorenz, S. Clark, I. Khan, and B. Coecke. Towards Compositional Interpretability for XAI, June 2024. URL http://arxiv.org/abs/2406.17583. arXiv:2406.17583 [cs, math].

J. van Eijck. Discourse Representation Theory. In K. Brown, editor, Encyclopedia of Language & Linguistics (Second Edition), pages 660–668. Elsevier, Oxford, Jan. 2006. ISBN 978-0-08-044854-1. doi: 10.1016/B0-08-044854-2/01090-7. URL https://www.sciencedirect.com/science/article/pii/B0080448542010907.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. arXiv:1706.03762 [cs], Dec. 2017. URL http://arxiv.org/abs/1706.03762. arXiv: 1706.03762.

G. Vilone and L. Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. Information Fusion, 76:89–106, Dec. 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2021.05.009. URL https://www.sciencedirect.com/science/article/pii/S1566253521001093.

V. Wang-Mascianica. String Diagrams for Text. DPhil, University of Oxford, 2023.

M. H. Waseem, J. Liu, V. Wang-Maścianica, and B. Coecke. Language-independence of DisCoCirc's Text Circuits: English and Urdu. Electronic Proceedings in Theoretical Computer Science, 366:50–60, Aug. 2022. ISSN 2075-2180. doi: 10.4204/EPTCS.366.7. URL http://arxiv.org/abs/2208.10281. arXiv:2208.10281 [cs].

J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. arXiv:1502.05698, 2015.

G. Wijnholds and M. Sadrzadeh. Evaluating Composition Models for Verb Phrase Elliptical Sentence Embeddings. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 261–271, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1023. URL https://www.aclweb.org/anthology/N19-1023.

G. Wijnholds and M. Sadrzadeh. A Type-Driven Vector Semantics for Ellipsis with Anaphora Using Lambek Calculus with Limited Contraction. Journal of Logic, Language and Information, 28(2):331–358, June 2019b. ISSN 1572-9583. doi: 10.1007/s10849-019-09293-4. URL https://doi.org/10.1007/s10849-019-09293-4.

M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural Language Questions for the Web of Data. In J. Tsujii, J. Henderson, and M. Paşca, editors, Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 379–390, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/D12-1035/.

R. Yeung and D. Kartsaklis. A CCG-Based Version of the DisCoCat Framework. arXiv:2105.07720 [cs, math], May 2021. URL http://arxiv.org/abs/2105.07720. arXiv: 2105.07720.

W. Zeng and B. Coecke. Quantum algorithms for compositional natural language processing. Electronic Proceedings in Theoretical Computer Science, 221:67–75, Aug. 2016. ISSN 2075-2180. doi: 10.4204/eptcs.221.8. URL http://dx.doi.org/10.4204/EPTCS.221.8.