VINCENT WANG-MAŚCIANICA

STRING DIAGRAMS FOR TEXT

2   v.w.

Contents

0
Synopsis

0.1   "What is this thesis about?"

THIS THESIS IS ABOUT A NEW WAY TO SEE THE SHAPE OF LANGUAGE USING FORMAL DIAGRAMS

THE DIAGRAMS LOOK LIKE CIRCUITS.
Let's say that the meaning of text is how it updates a model.
So we start with some model of the way things are.
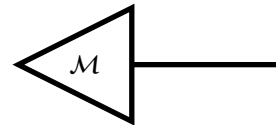
Text updates that model; like a gate updates the data on a wire.
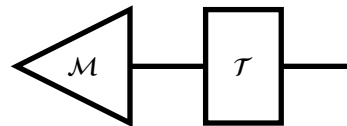
Text is made of sentences; like a circuit is made of gates and wires.
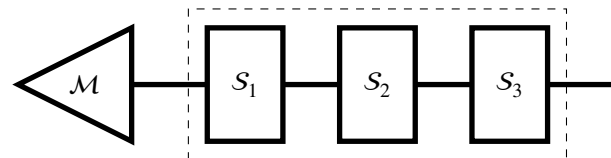
Let's say that ***The meaning of a sentence is how it updates the meanings of its parts.***
As a first approximation, let's say that the *parts* of a sentence are the nouns it contains or refers to.
Noun data is carried by wires.

Collections of nouns are related by gates.
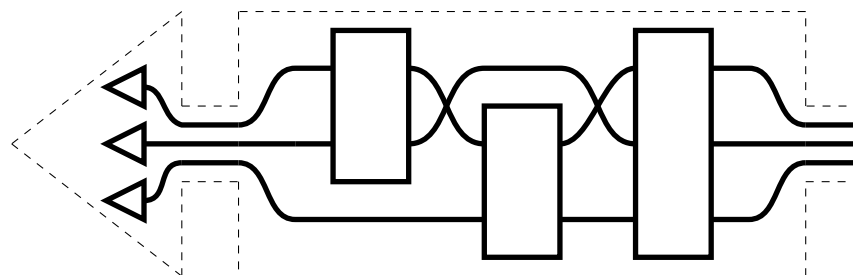
Gates can be related by higher order gates.
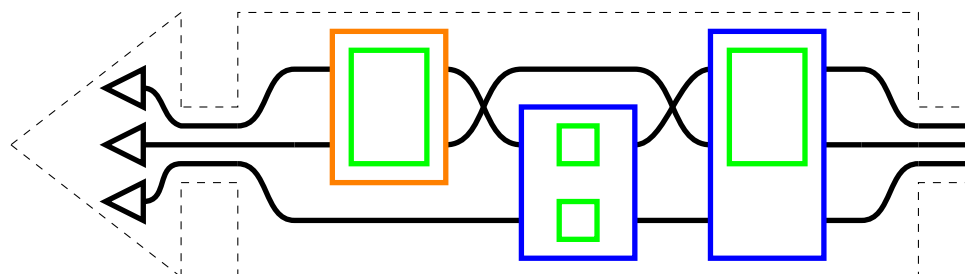
Higher order gates may be implemented as gates that modify the parameters of other gates.
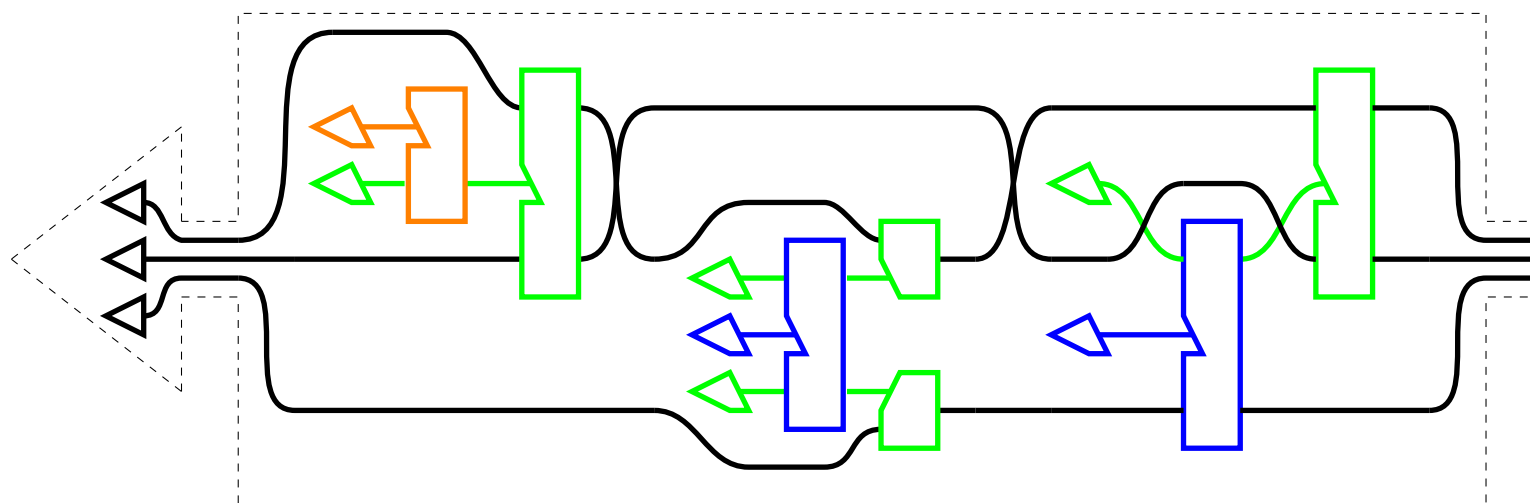
Every gate corresponds to a *content word* – a word with a dictionary meaning.

placeholder

Grammar, and *function words* – words that operate on meanings – are absorbed by the geometry of the diagram.

placeholder

Text diagrams are formal compositional blueprints.
These compositional blueprints may be instantiated by classical or quantum computers.
We know how grammar composes meanings in language.
We can quotient out grammar using these formal diagrams.
It turns big black boxes into composites of small black boxes,
which leaves smaller pieces for machines to learn representations for.
We introduce the mathematics, history, and philosophy of these diagrams in Chapter 1.

0.2   *"What is the shape of language?"*

Text circuits make different ways of thinking about language look the same
Category Theory is good at defining and bookkeeping structure.
Chapter 2 is about formal correspondences between different shapes of language.
We show how text circuits bridge several existing formalisms of language.

And we recount a proof of the expressive capacity of text circuits.
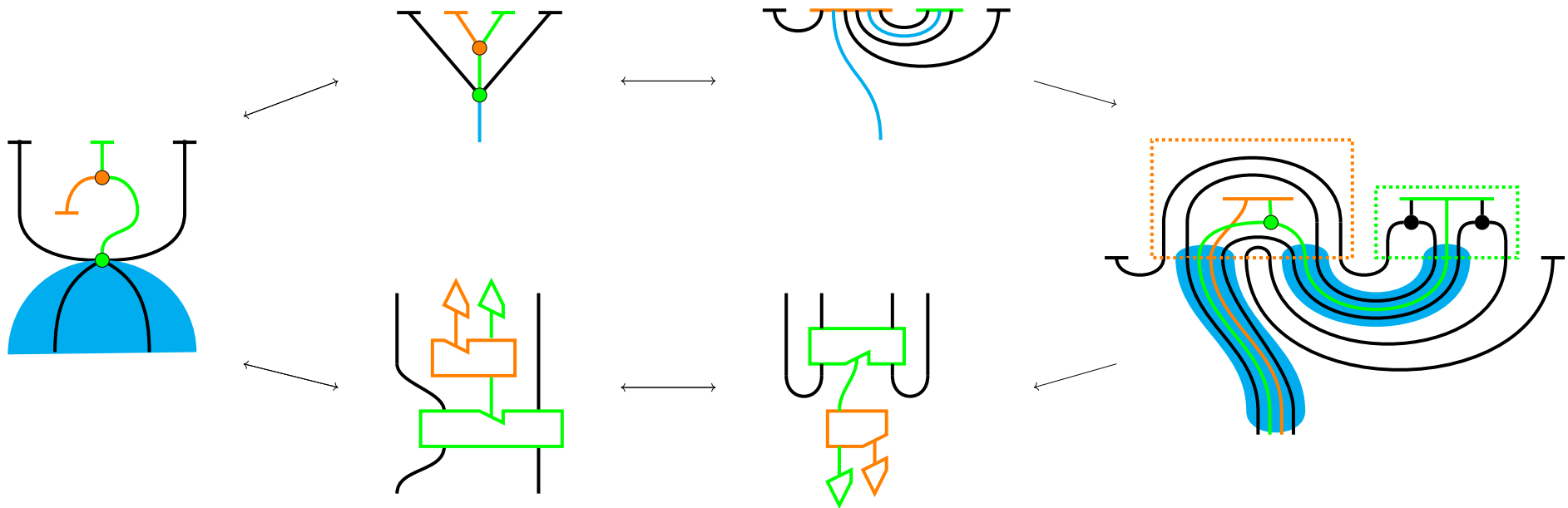
DISCOURSE REPRESENTATION THEORY
Text is composed of sentences as circuits are composed from gates.
Text circuits are composed by connecting noun wires.
This can happen when two sentences refer to the same noun.
`Alice likes Bob.  Bob hates Claire.`

placeholder

Or when a pronoun is used to refer to another noun. `Alice likes the flowers that Bob gives Claire.`

placeholder

So if you know how to resolve pronoun references,
and you know how to represent sentences as gates acting on noun wires,
then you may represent text as circuits.
There are several ways to obtain gates from grammars for sentences.

CONTEXT-FREE AND TREE-ADJOINING GRAMMARS

In a context-free grammar, a sentence symbol S is expanded by replacing individual symbols with strings of symbols to obtain a well-formed sentence.

The shape of these expansions is a tree.

That tree gives the shape of gates in a sentence.
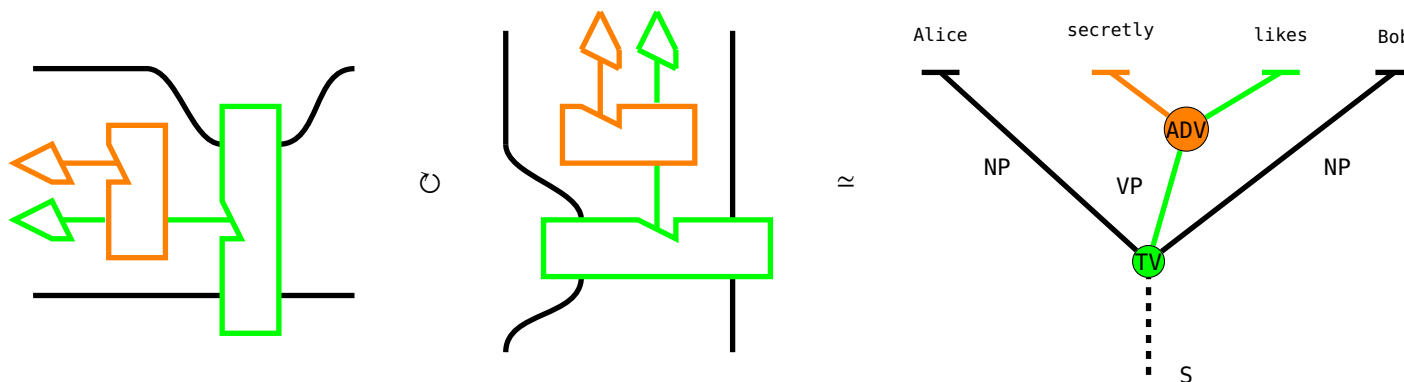
In Chapter **??**, we characterise the generative grammar of text circuits in terms of a context-free grammar with additional structure.

TYPELOGICAL GRAMMARS

A typelogical grammar is like the inverse of a context-free grammar.

The latter is the grammar of the speaker: a full sentence is produced from S.

The former is the grammar of the listener: the sentence type s must be derived from the words of the sentence.

In a typelogical grammar such as a pregroup grammar, words are assigned types.

These types come with reduction rules, that combine the types of multiple words.

A sequent calculus proof using the reduction rules witnesses the sentence type s:

$$
\cfrac{
  \text{Alice} : n \quad
  \cfrac{
    \cfrac{
      \text{secretly} : (^{-1}n \cdot s \cdot n^{-1}) \cdot (^{-1}n \cdot s \cdot n^{-1})^{-1} \quad \text{likes} :^{-1} n \cdot s \cdot n^{-1}
    }{
      \text{secretly\_likes} :^{-1} n \cdot s \cdot n^{-1}
    } [x^{-1} \cdot x \to 1]
  }{
    \text{Alice\_secretly\_likes} : s \cdot n^{-1}
  }
  \quad \text{Bob} : n
}{
  \text{Alice\_secretly\_likes\_Bob} : s
} [x \cdot^{-1} x \to 1]
$$

Such proofs of syntactic correctness can be re-expressed as diagrams:
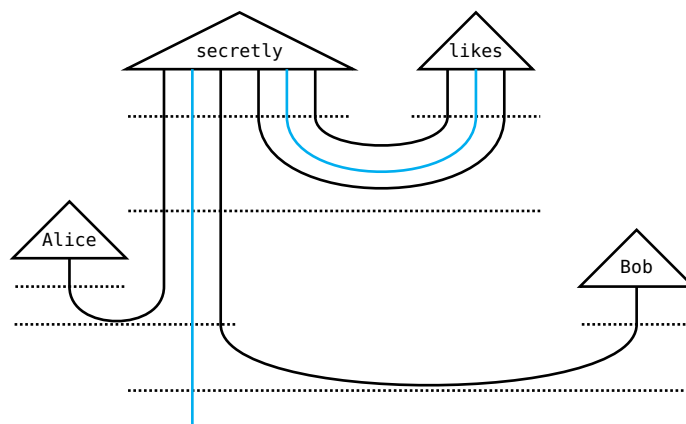
To obtain a circuit, we make the sentence type *s* dependent on the words that occur in the sentence.
To do this, we refine the structure on each word by introducing *internal wirings* for word-states.
Internal wirings make use of multiwires called spiders.
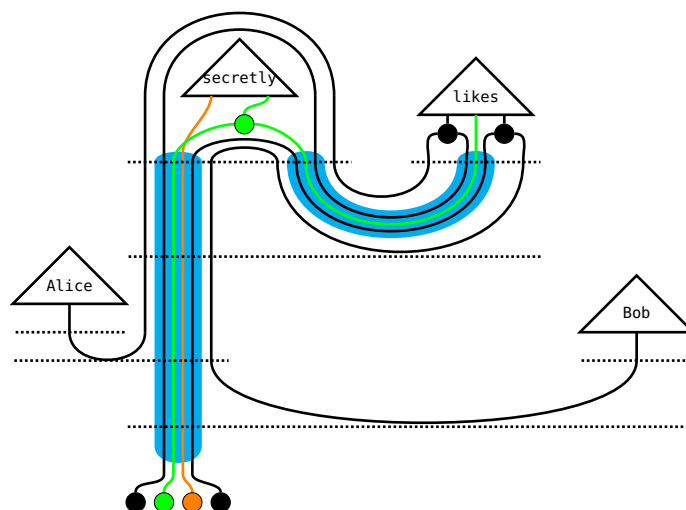
A certain choice of spider make these internal wiring diagrams topologically equivalent to the circuits we want.
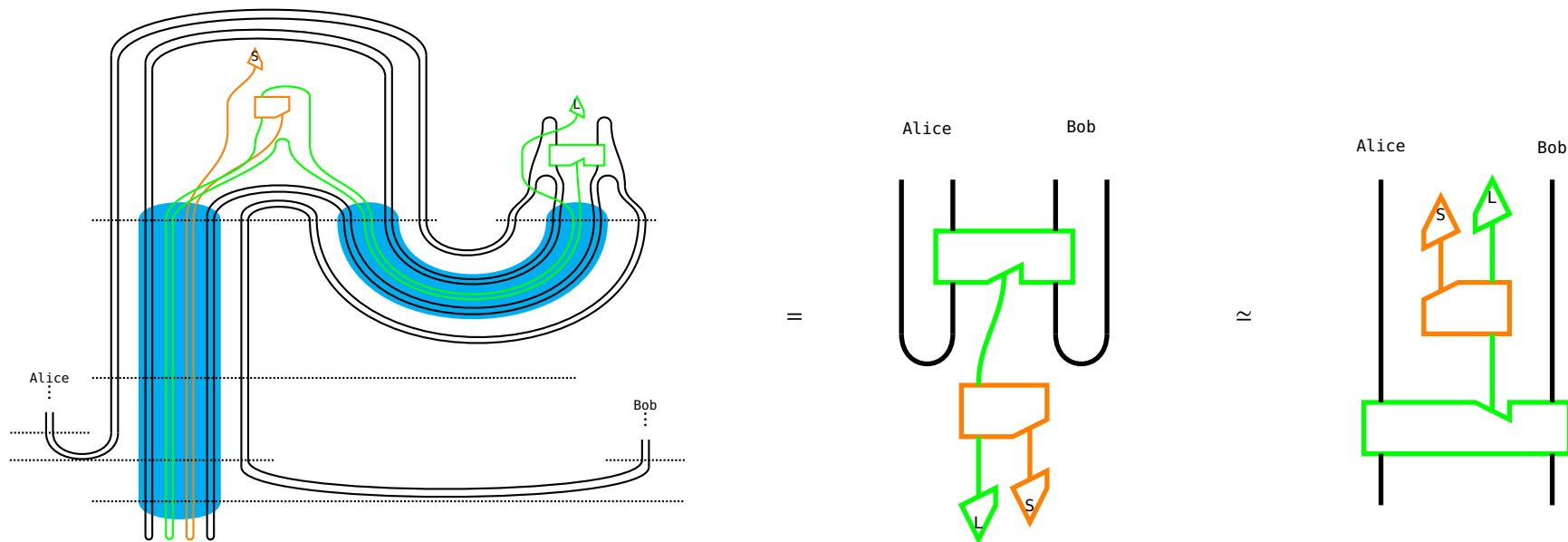
TEXTS WITH THE SAME MEANING BECOME THE SAME CIRCUIT!

placeholder

0.3   *"What is this thesis not about?"*

This thesis is about a particular idealised and computer-friendly conception of natural language syntax and semantics, stated using the as-of-yet uncommon mathematical dialect of applied category theory, and informed by natural language but bound in intent to empirical capture. This means there is no serious consideration of phonetics, phonology, morphology, pragmatics, and the historical, social and psychological dimensions of language. Text circuits do not provide grammaticality judgements upon one-dimensional strings of language, so in some sense they are not even a theory of syntax. There is, moreover, no correct way to instantiate the abstract gates of a text circuit, so in some sense they are not even a theory of semantics.[1]

While the application of this mathematical perspective to the puzzles of natural language is relatively fresh, there are no new mathematical techniques developed here. Nominally the most complicated mathematical gadgetry used

[1] So how are text circuits about syntax and semantics? Text circuits are a "natural" metasyntax for natural language. We start from two assumptions. First, that linguistic meanings are compositional – after all, we have finite dictionaries for words but not for sentences, and we can understand infinitely many novel sentences – and second, that the syntax of a sentence (whatever that might be) directs the composition of the meaning of the words. In Section 1.4.2, we see that taking syntax to be operadic (tree-shaped) and composition to be via lambda calculus,

will be finitely presented associative n-categories, but they are mostly useful to us as a combinatorial handle on symmetric monoidal categories with regions and rewrites, in the same way one would use a PROP with relations for just symmetric monoidal categories.

There will be no code, no demonstrations, nothing practical. I am only concerned with showing that certain things are achievable in principle here, and how they may be achieved. So I am taking refuge under the banner of "fundamental research", and for the usual reason: I can think of no demonstration that can outshine the state-of-the-art.

0.4   *"I'm a no-nonsense practical person, tell me about real-world impact."*

EXPRESSING GRAMMAR AS COMPOSITION OF PROCESSES MAY YIELD PRACTICAL BENEFITS.
Simplified, half of the problem of learning language is learning the meaning of words. The meanings change over time and are context-dependent, and the words are always increasing in number. Encoding these meanings by hand is a sisyphean task. Data-driven learning methods are a good fit: the patterns to be learnt are complex and nebulous, and there is a lot of data. However, data-driven methods may be weaker at the second half of the problem: learning how the rules of syntax work to compose meanings. It is difficult to do justice to compositionality in any practical setting, and the issue with data-driven architectures is either that we know immediately that they cannot be compositional, or their innards are too large and their workings too opaque to tell with confidence. Unfortunately, the issue of compositionality is a Maginot line erected by classical linguists and philosophers against the growing achievements of data-driven methods. It is a form of alienation we must all become accustomed to: that a lifetime's work is made irrelevant by a pile of linear algebra with an internet connection. Returning to the issue, I hope this framework can be a bridge, a way to split the cake fairly between the two halves of the problem: meanings for the machines, compositionality for the commons. Syntax is still difficult and quite vast, but the rules are finite and relatively static. We can break the black-box by reexpressing syntax as the composition of smaller black-boxes. We all stand to benefit: we may give machines an easier time – now they only have to learn the meanings of words well – and we can have confidence that the internal representations of the machine – their "mind's eye" – contains something we can probe and understand.

EXPRESSING GRAMMAR AS COMPOSITION OF PROCESSES IS A STEP TOWARDS BETTER HUMAN-MACHINE RELATIONS.
There is something fundamentally inhuman and behavioural about treating the production of language as a string of words drawn from a probability distribution. In practice, this is what large language models (LLMs) do. When *you* use language, do you feel like a diceroll? Even if we grant that the latent space of a data-driven architecture is

an analog for the space of internal mental states of a human user of language, how can we know whether the spaces are structurally analogous to the extent that human-machine partnership through the interface of natural language is safe? So here again is the possible solution: we can guarantee that the latent-space representation of the machine is built up in the same way we build up a mental representation when we read a book or watch a film. At the end of this chapter, after I have introduced the relevant mathematical formalisms, I'll show how we may build this bridge.

0.5   "I know a thing or two about language already, why should I read this?"

I WOULDN'T WANT TO BE A FORMAL LINGUIST TODAY. What would linguistics look like if it began today? LLMs would appear to us as oracles; wise, superhumanly capable at language, but inscrutable. Similar to how most people effortlessly use language without a formal understanding which they can express. So the fundamental mystery would remain: we would still seek to understand language, because understanding LLMs is completely different from understanding language[2]. So the presence of LLMs should not totally discourage our endeavour to understand language, but their sheer ability to do impressive things presents strong constraints about the acceptability of theories of language, by the usual standards of veridicality and utility[3]. In constast to the idealised and partial nature of formal theories, the nature of LLMs is that they are trained on empirical data about language that captures the friction of the real world. So, in terms of raw predictive power, we should naturally expect the LLMs to have an advantage over principled theories[4]. Here is a problem: if the better theory is one that lets you do more things, many theories of language are knocked out of the game by an LLM before they can even get started.

THERE ARE OTHER, INNATE, TENSIONS TO PLAYING THE GAME OF FORMAL LINGUISTICS.

I struggle to reconcile the empirical and scientific character to the study of language with the lack of usefulness; evidence of the success of the physical sciences are all around us, but where are the practical applications of formal linguistics? I have certainly felt like I was doing a kind of empirical observation by consulting my own acceptability judgements and intuitions when constructing models of English grammar. However, the laws and structure of language are smothered by a nebulous foliage of "unless" and "it depends" and other hedges. Probing for the elusive empirical truth of language by the objective process of asking others for grammaticality judgements helps a little, until you start to agree all the time, at which point there is a suspicion we could be fooling ourselves. To summarise: nobody can be sure whether they are truly isolating the essence of language or bullshitting, and to top it off, nowadays either case is irrelevant in practice! I hope to have convinced you of the need for meaningful standards to pursue formal linguistics by, in these depressing circumstances. I have been tempted to give up and treat the activity of formal linguistics as a form of art; harmless doodling for its own sake. But theorybuilding is distinguishable from fiction by other standards beyond veridicality and utility. I have tried to hold myself to the following standards,

---

[2] Suppose you knew the insides of a mechanical calculator by heart. Does that mean you *understand* arithmetic? At best, obliquely. Implementing ideal arithmetic means compromises; the calculator is full of inessentialities and tricks against the constraints of physics. You would not know where the tricks begin and the essence ends. Suppose you knew every line of code and every piece of data used to train an LLM. How could one delineate what is essential to language, and what is accidental?

[3] The explanatory value of a theory by itself – the "aha!" – is comparable to the aesthetic pleasure of art, but we can make up such stories all day. Explanatory value paired with veridicality and utility is the gold standard.

[4] Just as the Ptolemaic geocentric model of the solar system was more empirically precise than the heliocentric Copernican, though the latter was "more correct". This was because Ptolemaic epicycles can overfit to approximate any observed trajectory of planets. It took until Einstein's relativity to explain the perihelion of mercury, which at last aligned theoretical understanding with empirical observation.

which I think are reasonable:

- **If you can't beat them, make room for them.** There is value in synthesis. If your theory of language can neither empirically outperform nor work in tandem with neural methods in principle, it is a practical nonstarter.

- **If you can't beat them, be simpler.** There is value in a theory that provides unified understanding even if it is simplified, practically yet unrealised, or momentarily empirically embarrassed.

- **If you can't beat them, play a different game.** There is value in breaking new ground and extending our reach. If your theory of language can't make room and isn't simple and also has no force of originality beyond the reach of an LLM, it is boring.

- **If you can't beat them, smile and look pretty.** There is value in art. But if your theory of language is practically inferior, complicated, does nothing new, and on top of that it's *ugly*? You don't have a linguistic theory, you have a conspiracy theory.

I ARGUE THAT MOST FORMAL LINGUISTS HAVE A METALANGUAGE PROBLEM.

Set-theoretical foundations of mathematics are not well suited for complex and interacting moving parts. The chief drawback is that if you want to specify a function, you have to spell out how it behaves on the domain and codomain, which means spelling out what the innards of the domain and codomain are; to specify a set-theoretic model necessitates providing complete detail of how every part looks on the inside[5]. As you may be familiar with, this is a nightmare when dealing with a complex system: you have to specify all the implementation details from start to finish, bottom-up. This leads to at least three problems. First, the sociological problem is that this makes things difficult to understand unless you have invested a lot of time into mathematics. Second, interoperability is tricky. When a programmer wants to use a data structure or algorithm, they do not always write it from scratch or copy code from stackoverflow; they may use a library that provides them structures and methods they can call without worrying about how those structures and methods are implemented all the way down. However, if you building a complex theory by spelling out implementations set-theoretically from the start, incorporating a new module from elsewhere becomes difficult if that module has encoded things in sets differently. A lot of busywork must go into translating foundations of formalisms at an analogous level to machine code, which is time better spent building upwards. A computer scientist might say that some abstraction is needed, and being one, I say so. Third, and related to the second, is that set-theory is not the native language for the vast majority of practical computation. Often in the design of complex theories, we do not care about how precisely representations are implemented, instead we only care about placing constraints or guarantees on the behaviour of interacting interactions – that is, we care about operational semantics.

[5] This is a foundational, innate problem of set theory. Consider the case of the cartesian product of sets, one of the basic constructions. $A \times B$ is the "set of ordered pairs" (a, b) of elements from the respective sets, but there are many ways of encoding that are equivalent in spirit but not in syntax; a sign that the syntax is a hindrance. What we really want of the product is the property that $(a, b) = (c, d)$ just when $a = c$ and $b = d$. Now here is a small sampling of different ways to encode an ordered pair. Kuratowski's definition is

$$A \times B := \left\{ \{\{a\}, \{a, b\}\} \mid a \in A \, , \, b \in B \right\}$$

Which could have just as easily been:

$$A \times B := \left\{ \{a, \{a, b\}\} \mid a \in A \, , \, b \in B \right\}$$

And here is Wiener's definition:

$$A \times B := \left\{ \{\{a, \varnothing\}, b\} \mid a \in A \, , \, b \in B \right\}$$

IF I HAD TO BE A FORMAL LINGUIST TODAY, I WOULD USE APPLIED CATEGORY THEORY.

A broad theme of this thesis is to illustrate the economy and reach of applied category theory for dealing with compositional phenomena. Our capacity for language is one of the oldest and sophisticated pieces of compositional technology, maybe even the foundation of compositional thought. So, linguists are veteran students of compositionality and modularity. How does syntax compose meaning? How do the constraints and affordances of language interact? The discipline embodies a encyclopaedic record of how compositionality works "in the field", just as botanists record flowers, early astronomers the planetary motions, or stamp collectors stamps. But a disparate collection of observations does not a theory make; we will inevitably wish to bring it all together, one way or another. The problems I have mentioned above are obstacles, and I hope to show that using applied category theory as a metalanguage may be a solution. We may work with formal diagrams that our visual cortexes have built-in rules to manipulate, and we are free to work at the level of abstraction we choose, so that we may easily incorporate other modules and find implementations in a variety of settings. Later in this chapter, we introduce these concepts again via a worked example of getting and putting values into a database using a family of operations bound by equational relations, and the consequence of this view is that it does not matter whether the operations are realised using pencil and paper, bits, qubits, or anything else; as long as they obey our equational constraints, they will behave precisely as getters and putters for a database entry.

0.5.1   "What's wrong with the Montagovian λ-calculus approach to compositionality?"

Here is a fair challenge from a hypothetical formal semanticist: *"We already have a general-purpose metalanguage solution for composition: the λ-calculus. What can we do with category theory that we cannot already do?"* Shortest answer: Punchcard machines are turing-complete, so the answer is that there is nothing that a high-level language like python on a modern laptop can compute that a punchcard machine cannot, and we are all free to program how we like. A more constructive and historically-situated answer is in Section 1.4. The λ-calculus is about plugging expressions into contexts of other expressions, and the Montagovian approach is the observation that you can match the shape of how you plug things together with the shape of syntax, therefore obtaining semantics composed out of grammar. To be clear, we are staying in the business of information-plumbing, and we are keeping Montague's core insight that there is a homomorphism – a structure preserving map – from syntax to semantics. The difference lies in formalising these homomorphisms as functors, which allows us to replace the λ-calculus with other, more expressive playgrounds on the syntax and semantics ends. This merits an immediate illustration. Consider that language is one-dimensional as an artefact of the spoken word coming before the written. On the other hand, meanings – whatever they are – are definitely not shaped like lists of words, they have more structure to them. So the relation between syntax and semantics can be viewed as the solution to the problem of encoding and decoding complex structured meanings as one-dimensional strings. There cannot be a unique solution to the problem

of encoding and decoding, or else our internal mental lives would be isomorphic to lists of words. So – as we can see from all the natural languages in the world – there are multiple solutions, multiple ways of encoding the same meaning as a string of words and decoding them to get back the same meaning. We find such redundancy even in the same language. For example, if I say that `Alice likes the flowers that Bob gave Claire.`, I would ideally like a grammatical equation telling me that my utterance means the same thing as something like `Bob gave Claire flowers.  Alice likes those flowers.` But if all meanings are modelled as single-output functions and function-application, both utterances will have the same featureless meaning type – there is no foothold to write an equation there. So instead perhaps:

$$[\![\texttt{that}]\!] := \lambda\ \mathsf{N}_1\ \mathsf{N}_2\ \mathsf{N}_3\ \mathsf{N}_4\ \mathsf{TV}\ \mathsf{DV}\ .\ \ \mathsf{TV}(\mathsf{N}_1, \mathsf{N}_2) \wedge \mathsf{DV}(\mathsf{N}_3, \mathsf{N}_4, \mathsf{N}_2)$$

A problem persists. What if I continue uttering some story about Alice and Bob and Claire and only much later mention the flowers again? It would be a lot of trouble to devise a system of combinators like the one above that keeps everything organised. The usual solution to this train of thought – as we will see in Section **??**, rediscovered independently in logic, computer science, relativity theory, and quantum mechanics – is to power up function expressions to give multiple outputs using a system of indices. This way we can use indices to keep track of `the flowers` and plug them in where they are needed later on in the text. The usual observation that follows after this trick is that sequential composition of two pairs of parallel functions is equivalent to the parallel composition of two pairs of sequentially composed functions, i.e., writing parallel composition as "$\otimes$" and sequential composition as ";", we have:

$$(f_i \otimes g_j); ({}^i h \otimes^j k) = (f_i; {}^i h) \otimes (g_j; {}^j k)$$

This rule, like the meanings behind language, does not want to live on a one-dimensional string. When we find the right place for expressions to live, the bureaucracy of syntax melts away, absorbed by the geometry of the medium. The mathematics of sequential and parallel composition wants to live in two dimensions, like this:

placeholder

Applied category theory for diagrams, put another way, is the mathematics of exploiting this connection between algebra and geometry. We can express syntax and semantics in monoidal categories, and relate them as Montague would by a monoidal functor. The (typed) λ-calculus corresponds to a specific kind of monoidal category, a cartesian closed one, whereas most practical things live in the more general setting of symmetric monoidal categories. Later in this chapter, I will show you how it is done.

0.6   *"So what can you do that we couldn't already? What are the novel contributions?"*

I DON'T CLAIM ORIGINALITY FOR CONTENT BUT... I also do not care if anything I present is a rediscovery, which is inevitable because breadth is the point; the real trick is that I use the same small handful of formal tools as a unified approach. All I have done is a kind of intellectual arbitrage by applying simple methods from one field in another. Here's a list of what's possibly new:

- A brief prehistory of string diagrams.

- A brief characterisation of Montague's conception of syntax in modern mathematical terminology.

- A theory of text structure compatible with quantum and classical data-driven modelling.

- A structural correspondence between tree-adjoining grammars, typelogical grammars, and discourse representation theories.

- A recapitulation of the characterisation of text circuits by a controlled fragment of English from [longpaper], this time including the formal n-categorical signatures.

- An investigation of a category of continuous relations, culminating in a string-diagrammatic characterisation of basic linguistic topological concepts.

- Formal semantics for interacting spatially-embodied agents and higher-order predication.

- A formal correspondence between monoidal computer [] and a generalisation of anaphora beyond nouns and events.

- A method to formally model and compute textual metaphors.

- A linguistic characterisation of o-minimal structures.

To elaborate, in Chapter 2, I will explain a small miracle, that certain generative grammars that produce sentences and typelogical grammars that deduce syntactic correctness – the grammars of speech and listening – don't just produce the same set of sentences, they are structurally related to the bone; as they must be, or else we would be unable to transmit meanings by encoding and decoding in language. Further, I show how symmetric monoidal structure is a natural setting to interpret higher-order predicates – put simply, the manner in which `quickly` modifies the word `runs` can be viewed as the action of a process that modifies a parameter of another process. After setting a formal stage to perform calculations with basic linguistic spatial relations in Chapter 3, In Chapter 4 I will; escape

the confines of truth-conditional semantics to give formal semantics to the conduit metaphor specifically and textual metaphor in general, thus making sense of utterances like `to put an idea into words`; provide an elementary and possible-worlds-free recipe to formalise the intensional semantics of words such as `want` in terms of containers; provide a toy mathematical setting to generalise anaphora beyond nouns and events to any meaningful component of text; bridge natural language to foundational mathematics by outlining formal relationship between intuitive "tame topologies", computational learning theory, and spatial language via text circuits for o-minimal structures. All this is to suggest that...

... THIS WAY OF RECKONING WITH LANGUAGE PUTS A UNIFIED THEORY OF LANGUAGE WITHIN REACH. Maybe. I am young and naïve enough at the time of writing to be a little dramatic, a little hopeful for the future. In truth I would be deeply unhappy if any of what I write is taken too seriously and rigidly, without a sense of play; I want to demonstrate how mathematics can be used to create and explore meaning rather than subjugate it. I would be happy and consider this thesis a success if you, reader, simply enjoyed my diagrams for their aesthetic value alone. I would be marginally happier if I manage to touch you in a way that changes how you look at language.

0.7   Why is this thesis landscape?

Why should it be otherwise? A lot of screens are landscape. My diagrams are needy of space, and it is nice to be able to read them from left to right like text. The fat margins are good for citations, and asides both formal and informal.

1
Background

1.1   A Partial History of String Diagrams

OBSERVATION 1: WE HAVE A PRIMITIVE URGE TO DEPICT.

cavedrawingstolanguage

OBSERVATION 2: NATURAL LANGUAGE PRESENTS ITSELF AS A UNIVERSAL INTERFACE FOR THOUGHT.

placeholder

THE PICTORIAL CAPTURE OF LANGUAGE AND THOUGHT IS A CURSED DREAM. We suffer a seductive alchemical urge to capture the essence, laws and structure of things in words and pictures – containment in symbol and form. On occasion, this urge is focused on itself as an ouroboros; a doomed endeavour to use the faculties of abstraction, representation, and structure to consume itself.

peirce

magnusson

maybejung

IT IS OUR TURN TO SUFFER THIS SICKNESS. The difference this time around is in approach: we use *string diagrams*. If depiction is the domain of geometry – the mathematics of form in space, and if language is the domain of algebra – the mathematics of symbolic manipulation, then string diagrams are our bridge. We recall a quote by the late Michael Atiyah:

Algebra is the offer made by the devil to the mathematician. The devil says: "I will give you this powerful machine, it will answer any question you like. All you need to do is give me your soul: give up geometry and you will have this marvellous machine."

String diagrams are a loophole in the devil's contract. Descartes' led an incursion into geometry with algebra, and string diagrams are the belated counteroffensive, or peaceful synthesis.

STRING DIAGRAMS ARE FORMAL REPRESENTATION AND REASONING SYSTEMS FOR MONOIDAL CATE-GORIES. "Formal" distributes over "representation" and "reasoning", which arose chronologically in that order.

1.1.1   Formal visual representation

Pre-formal diagrams are about visual expression. By the restrictions of writing technologies, the kind of visual representations we are interested in are two-dimensional. So visual means geometric in the plane. The ancient ancestors of modern string diagrams are systems for visual representation of formal symbolic constructions. For these ancestral diagrams, reasoning, if there is any, takes place elsewhere in the symbolic realm, not within and between diagrams.

$$Euclid : Geometry$$

$$Venn\&Carroll : Syllogisms$$

$$Petri : Chemistry$$

1.1.2   Convergent Evolution

String diagrams arise naturally when keeping track of many pairwise connections between entities becomes unwieldy for one-dimensional syntax. The usual evolutionary path is to first adopt Einstein notation with indexed superscripts and subscripts for bookkeeping, followed by the depiction of connected indices as wires in the plane.

IN PHYSICS

IN LOGIC

IN COMPUTER SCIENCE

1.1.3   Formal visual reasoning

1.2   Process Theories

A *process* is something that transforms some number of input system types to some number of output system types. We depict systems as wires, labelled with their type, and processes as boxes. We read processes from left to right.

Inputs   Process   Outputs

A

B

Φ

C

D

E
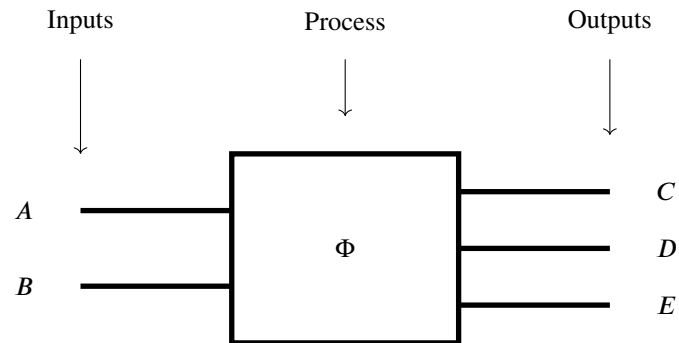
Processes may compose in parallel, which we depict as vertically stacking boxes.

A

B

Φ

C

D

E

E

C

Ψ

A
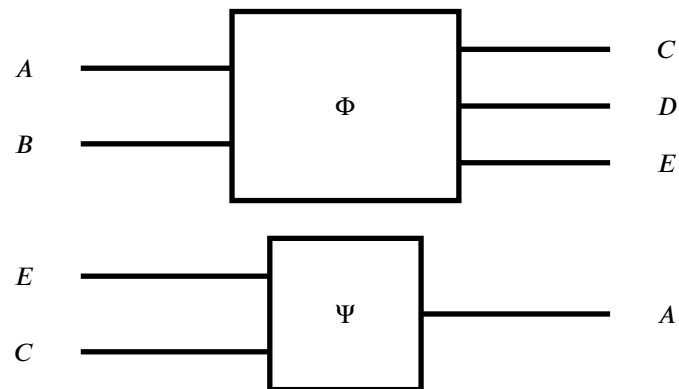
Processes may compose sequentially, which we depict as connecting wires of the same type from left to right.

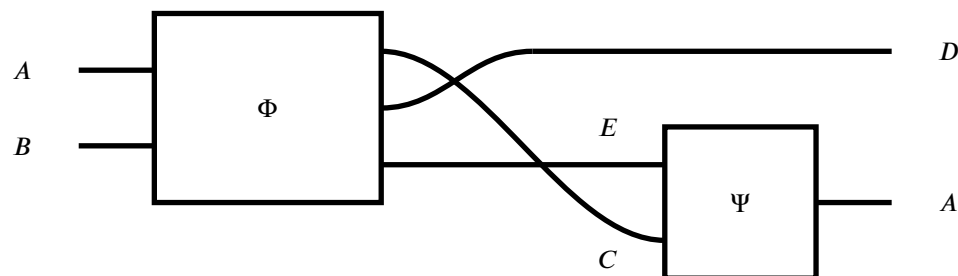In these diagrams only input-output connectivity matters: so we may twist wires and slide boxes along wires to obtain different diagrams that still refer to the same process. So the diagram below is equal to the diagram above.
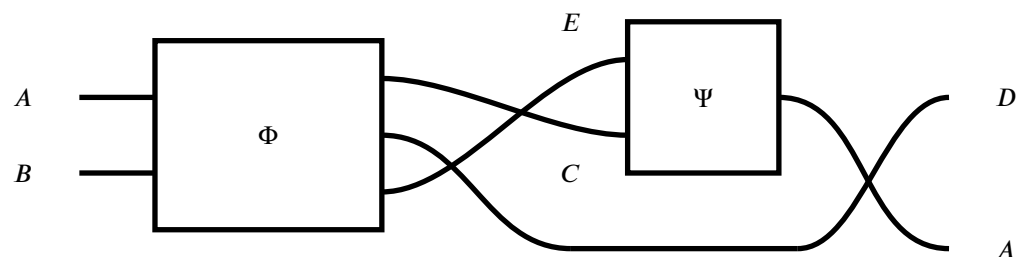
Some processes have no inputs; we call these *states*.
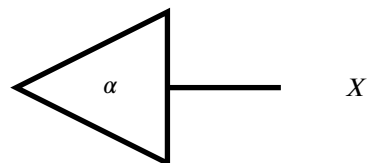
Some processes have no outputs; we call these *tests*.

A process with no inputs and no outputs is a *number*; the number tells us the outcome of applying tests to a composite of states modified by processes.

A process theory is given by the following data[1]:

- A collection of systems

- A collection of processes along with their input and output systems
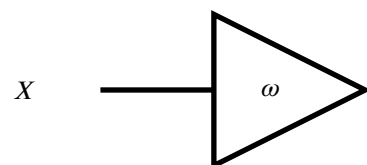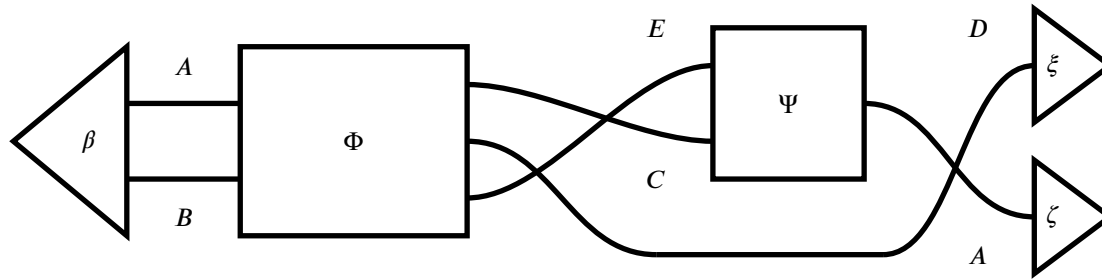
- A methodology to compose systems and processes sequentially and in parallel, and a specification of the unit of parallel composition.

- A collection of equations between composite processes

Example 1.2.1 (Linear maps with direct sum). Systems are finite-dimensional vector spaces over $\mathbb{R}$. Processes are linear maps, expressed as matrices with entries in $\mathbb{R}$.

Sequential composition is matrix multiplication. Parallel composition of systems is the direct sum of vector spaces $\oplus$. The parallel composition of matrices $\mathbf{A}, \mathbf{B}$ is the block-diagonal matrix

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$$

The unit of parallel composition is the singleton 0-dimensional vector space. States are row vectors. Tests are column vectors. The numbers are $\mathbb{R}$.[2]

Example 1.2.2 (Sets and functions with cartesian product). Systems are sets A, B. Processes are functions between sets $f : A \to B$. Sequential composition is function composition. Parallel composition of systems is the cartesian product of sets: the set of ordered pairs of two sets.

$$A \otimes B = A \times B := \{(a, b) \mid a \in A, b \in B\}$$

[1] Formally, process theories are symmetric monoidal categories []; see section **??**.

[2] Usually the monoidal product is written with the symbol $\otimes$, which denotes hadamard product for linear maps. The process theory we have just described takes the direct sum $\oplus$ to be the monoidal product. To avoid confusion we will use the linear algebraic notation when linear algebra is concerned.

The parallel composition $f \otimes g : A \times C \to B \times D$ of functions $f : A \to B$ and $g : C \to D$ is defined:

$$f \otimes g := (a, c) \mapsto (f(a), g(c))$$

The unit of parallel composition is the[3] singleton set $\{\star\}$. States of a set A correspond to elements $a \in A$.[4] Every system A has only one test $a \mapsto \star$.[5] There is only one number.

Example 1.2.3 (Sets and relations with cartesian product). Systems are sets A, B. Processes are relations between sets $\Phi \subseteq A \times B$, which we may write in either direction $\Phi^* : A \nrightarrow B$ or $\Phi_* : B \nrightarrow A$.[6] Sequential composition is relation composition:

$$A \overset{\Phi}{\nrightarrow} B \overset{\Psi}{\nrightarrow} C := \{(a, c) \mid a \in A, \ c \in C, \ \exists b_{\in B} : (a, b) \in \Phi \wedge (b, c) \in \Psi\}$$

Parallel composition of systems is the cartesian product of sets. The parallel composition of relations $A \otimes C \overset{\Phi \otimes \Psi}{\nrightarrow} B \otimes D$ of relations $A \overset{\Phi}{\nrightarrow} B$ and $C \overset{\Psi}{\nrightarrow} D$ is defined:

$$\Phi \otimes \Psi := \{\big((a, c), (b, d)\big) \mid (a, b) \in \Phi \wedge (c, d) \in \Psi\}$$

The unit of parallel composition is the singleton. States of a set A are subsets of A. Tests of a set A are also subsets of A.

1.2.1   What does it mean to copy and delete?

Now we discuss how we might define the properties and behaviour of processes by positing equations between diagrams. Let's begin simply with two intuitive processes *copy* and *delete*:

Example 1.2.4 (Linear maps). Consider a vector space $\mathbf{V}$, which we assume includes a choice of basis. The copy map is the rectangular matrix made of two identity matrices:

$$\Delta_{\mathbf{V}} : \mathbf{V} \to \mathbf{V} \oplus \mathbf{V} := \begin{bmatrix} \mathbf{1}_{\mathbf{V}} & \mathbf{1}_{\mathbf{V}} \end{bmatrix}$$

The delete map is the column vector of 1s:

$$\epsilon_{\mathbf{V}} : \mathbf{V} \to \mathbf{0} := \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

[3] There are many singletons, but this presents no problem for the later formal definition because they are all equivalent up to unique isomorphism.

[4] We forgo the usual categorical definition of points from the terminal object in favour of generalised points from the monoidal perspective.

[5] This is since the singleton is terminal in **Set**.

[6] Relations between sets are equivalently matrices with entries from the boolean semiring. Relation composition is matrix multiplication with the boolean semiring. Φ^*, Φ_* are the transposes of one another.

coassociativity

Example 1.2.5 (Sets and functions). Consider a set A. The copy function is defined:

$$\Delta_A : A \to A \times A := a \mapsto (a, a)$$

The delete funtion is defined:

$$\epsilon_A : A \to \{\star\} := a \mapsto \star$$

cocommutativity

Example 1.2.6 (Sets and relations). Consider a set A. The copy relation is defined:

$$\Delta_A : A \nrightarrow A \times A := \{(a, (a, a)) \mid a \in A\}$$

The delete relation is defined:

$$\epsilon_A : A \nrightarrow \{\star\} := \{(a, \star) \mid a \in A\}$$

counitality   (1.1)

We may verify that, no matter the concrete interpretation of the diagram in terms of linear maps, functions or relations copy and delete satisfy the equations in the margin:

It is worth pausing here to think about how one might characterise the process of copying in words; it is challenging to do so for such an intuitive process. The diagrammatic equations in the margin, when translated into prose, provide an answer.

Coassociativity: says there is no difference between copying copies.

Cocommutativity: says there is no difference between the outputs of a copy process.

Counitality: says that if a copy is made and one of the copies is deleted, the remaining copy is the same as the original.

Insofar as we think this is an acceptable characterisation of copying, rather than specify concretely what a copy and delete does for each system X we encounter, we can instead posit that so long as we have processes $\Delta_X :$ $X \otimes X \to X$ and $\epsilon_X : X \to I$ that obey all the equational constraints above, Δ_X and ϵ_X are as good as a copy and delete.

Here is an unexpected consequence. Suppose we insist that *to copy* in principle also implies the ability to copy *anything* – arbitrary states. From Example 1.2.7 and Remark 1.2.8, we know that this demand is incompatible with certain process theories. In particular, this demand would constrain a process theory of sets and relations to a subtheory of sets and functions. The moral here is that process theories are flexible enough to meet ontological needs. A classical computer scientist who works with perfectly copyable data and processes might demand universal copying along with Equations 1.1, whereas a quantum physicist who wishes to distinguish between copyable classical data and non-copyable quantum data might taxonomise copy and delete as a special case of a more generic quasicopy and quasi-delete that only satisfies equations 1.1.[7]
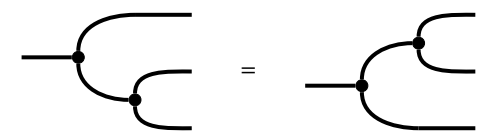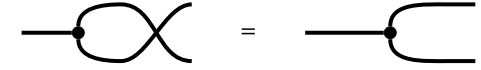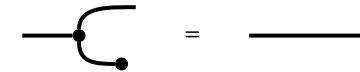
Example 1.2.7 (Not all states are copyable). Call a state *copyable* when it satisfies the following diagrammatic equation:
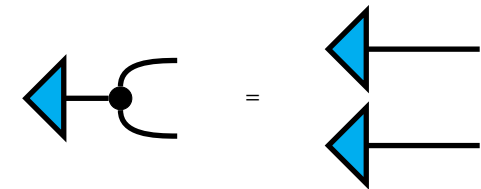
In the process theory of sets and functions, all states are copyable. Not all states are copyable in the process theories of sets and relations and of linear maps. For example, consider the two element set $\mathbb{B} := \{0, 1\}$, and let $\top : \{\star\} \nrightarrow \mathbb{B} := \{(\star, 0), (\star, 1)\} \simeq \{0, 1\}$. Consider the composite of $\top$ with the copy relation:

1.2.2   What is an update?

In the previous section we have seen how we can start with concrete examples of copying in distinct process theories, and obtain a generic characterisation of copying by finding diagrammatic equations copying satisfies in each concrete case. In this section, we show how to go in the opposite direction: we start by positing diagrammatic equations that characterise the operational behaviour of a particular process – such as *updating* – and it will turn out that any concrete process that satisfies the equational constraints we set out will *by our own definition* be an update.

Perhaps the most familiar setting for an update is a database. In a database, an entry often takes the form of pairs of fields and values. For example, where a database contains information about employees, a typical entry might look like:

<   **NAME**:Jono Doe,   **AGE**:69,   **JOB**:CONTENT CREATOR,   **SALARY**:$420,   ...   >

There are all kinds of reasons one might wish to update the value of a field: Jono might legally change their name, a year might pass and Jono's age must be incremented, Jono might be promoted or demoted or get a raise and so on. It was the concern of database theorists to formalise and axiomatise the notion of updating the value of a field – *independently of the specific programming language implementation of a database* – so that they had reasoning tools to ensure program correctness []. The problem is reducible to axiomatising a *rewrite*: we can think of updating a value as first calculating the new value, then *putting* the new value in place of the old. Since often the new value depends in some way on the old value, we also need a procedure to *get* the current value.

That was a flash-prehistory of *bidirectional transformations* []. Following the monoidal generalisation of lenses in [], a rewrite as we have described above is specified by system diagrammatic equations in the margin, each of which we introduce in prose.

These diagrammatic equations do two things. First, they completely specify what it means to get and put values in a field in an implementation independent manner; it doesn't matter whether database entries are encoded as bitstrings, qubits, slips or paper or anything else, what matters is the interaction of get and put. Second, the diagrammatic equations give us the right to call our processes *get* and *put* in the first place: we define what it means to get and put by outlining the mutual interactions of get, put, copy, and delete. These two points are worth condensing and rephrasing:

A *kind of process* is determined by patterns of interaction with other kinds of processes.

Now we can diagrammatically depict the process of updating Jono's age, by getting Jono's age value from their entry, incrementing it by 1, and putting it back in.
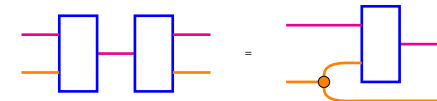
PutPut: Putting in one value and then a second is the same as deleting the first value and just putting in the second.
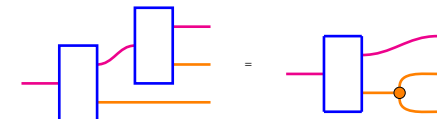

GetPut: Getting a value from a field and putting it back in is the same as not doing anything.
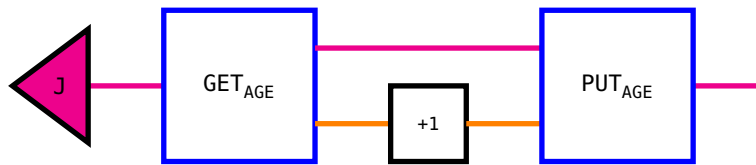

PutGet: Putting in a value and getting a value from the field is the same as first copying the value, putting in one copy and keeping the second.

GetGet: Getting a value from a field twice is the same as getting the value once and copying it.

1.2.3   Spatial predicates

The following simple inference is what we will try to capture process-theoretically:

- `Oxford is north of London`

- `Vincent is in Oxford`

- `Rocco is in London`

How might it follow that `Rocco is south of Vincent?`

One way we might approach such a problem computationally is to assign a global coordinate system, for instance interpreting 'north' and 'south' and 'is in' using longitude and latitude. Another coordinate system we might use is a locally flat map of England. The fact that either coordinate system would work is a sign that there is a degree of implementation-independence.

This coordinate/implementation-independence is true of most spatial language: we specify locations only by relative positions to other landmarks or things in space, rather than by means of a coordinate system. This is necessarily so for the communication of spatial information between agents who may have very different reference frames.

So the process-theoretic modelling aim is to specify how relations between entities can be *updated* and *classified* without requiring individual spatial entities to intrinsically possess meaningful or determinate spatial location.

So far we have established how to update properties of individual entities. We can build on what we have so far by observing that a relation between two entities can be considered a property of the pair.

placeholder

Spatial relations obey certain compositional constraints, such as transitivity in the case of 'north of':

placeholder

Or the equivalence between 'north of' and 'south of' up to swapping the order of arguments:

There are other general constraints on spatial relations, such as order-independence: the order in which spatial relations are updated does not (at least for perfect reasoners) affect the resultant presentation. This is depicted diagrammatically as commuting gates:

placeholder

1.2.4   *Some basic properties of linguistic spatial relations*

1.3   Defining String Diagrams

1.3.1   Symmetric Monoidal Categories

Definition 1.3.1 (Category). A *category* C consists of the following data

- A collection $\mathrm{Ob}(C)$ of *objects*

- For every pair of objects $a, b \in \mathrm{Ob}(C)$, a set $C(a, b)$ of *morphisms* from a to b.

- Every object $a \in \mathrm{Ob}(C)$ has a specified morphism 1_a in $C(a, a)$ called *the identity morphism* on a.

- Every triple of objects $a, b, c \in \mathrm{Ob}(C)$, and every pair of morphisms $f \in C(a, b)$ and $g \in C(b, c)$ has a specified morphism $(f; g) \in C(a, c)$ called *the composite* of f and g.

This data has to satisfy two coherence conditions, which are:

UNITALITY: For any morphism $f : a \to b$, $1_a; f = f = f; 1_b$

ASSOCIATIVITY: For any four objects a, b, c, d and three morphisms $f : a \to b, g : b \to c, h : c \to d$, $(f; g); h = f; (g; h)$

Definition 1.3.2 (Categorical Product).

Example 1.3.3 (Cartesian product of sets).

Definition 1.3.4 (Functor).

Definition 1.3.5 (Natural Transformation).

Definition 1.3.6 ($(1-)$**Cat**).

Example 1.3.7 (Monoids in **Set**).

Definition 1.3.8 (Monoidal Category).

Definition 1.3.9 (Symmetric Monoidal Category).

Theorem 1.3.10 (Strictification).

Theorem 1.3.11 (Graphical?).

1.3.2   PROPs

PROP stands for "**PRO**duct and **P**ermutations category", but it may as well be an acryonym for "**P**icture **R**ules **Of** **P**rocesses".

1.3.3   1-object 4-categories

(Typed) Lambda-Calculus	Intuitionistic Logic	Cartesian Closed Categories
Types	Propositions	Categories
Curry	Howard	Lambek

1.4 A brief history of formal linguistics from the categorial perspective

SUMMARY: Logical Type Theory in mathematics had a twin sister Categorial Grammar in linguistics, born in the 30s to Ajdukiewicz, raised into the 50s by Bar-Hillel and Lambek. Montague brought formal semantics to the picture via the Lambda-Calculus in the 70s, around the time Category Theory was getting started. The Curry-Howard correspondence between types and logic became Curry-Howard-Lambek to include categories, thus relating the typed lambda-calculus, intuitionistic logic, and cartesian closed categories. Lambek and Moortgat evolved categorial grammar into typelogical grammar; the use of different proof systems as models of grammar, which in turn suggested semantic categories beyond the cartesian closed setting. Alternative semantics in the form of quantum computers were realised by Coecke and Lambek, who together added string diagrams to the correspondence via a *literally* observed correspondence between quantum bell states and reductions in Pregroup Grammar. Sazradeh and Clark enter the collaboration, bringing in Firth's distributional semantics – which had by the time become computationally practical as the basis of neural methods for word encoding – to create DisCoCat; a Distributional, Compositional and Categorial framework for language. Mirroring the developmental circumstances of Discourse Representation Theory (itself independently conceived by Kamp and Heim), Coecke suggested promoting DisCoCat as framework for sentences towards a circuit-shaped framework for text – DisCoCirc. But there remained unanswered questions and ugly spots, and some poor sap had to work out the formal details and clean things up. That poor sap is me.

1.4.1 Curry-Howard-Lambek

This correspondence means that you can use the lambda-calculus on any family of data organised as a cartesian closed category; this could be strings, or sets and functions, topological shapes with holes, neural nets and finite vectors. So one small benefit of the category-theoretic viewpoint is a formal underpinning for these mild extensions.

Describing the bigger benefit requires a bigger picture.

and Lambek and Coecke fully integrated the picture with syntax and categories. So the linguist's trinity may look something like this:

Or this:

So here is the story today as far as a linguist may be concerned. We know that Curry-Howard-Lambek- correspondence generalises: if you poke Howard for a grammar that is expressively distinct from a CCG, the type-theory

Computation	Syntax	Semantics
(Typed) Lambda-calculus	Combinatory Categorial Grammar	Cartesian Closed Categories
Montague	Ajdukiewicz	Lambek

Computation	Syntax	Semantics
(Typed) Lambda-calculus	Pregroup Grammar	Rigid Autonomous Monoidal Categories
Montague	Ajdukiewicz	Lambek

changes, so Lambek gives a different family of semantic categories with different internal logics, and Curry gives you a syntactic composition gadget that differs from the lambda-calculus.

1.4.2   What did Montague consider grammar to be?

SUMMARY: Montague considered grammars to be coloured operads; Montague's "algebras" are (multi-sorted) clones, which are in bijection with (multi-sorted) Lawvere Theories, which are equivalently coloured operads.

MONTAGUE SEMANTICS/GRAMMAR AS MONTAGUE ENVISIONED IT IS LARGELY CONTAINED IN TWO PAPERS – *Universal Grammar* [8], and *The Proper Treatment of Quantifiers in English* [9] – both written shortly before his murder in 1971. The methods employed were not *mathematically* novel – the lambda calculus had been around since [DATE], and Tarski and Carnap had been developing intensional higher-order logics since [] – but for linguists who, by-and-large, only knew first order predicate logic, these methods were a tour-de-force that solved longstanding problems in formal semantics. Thus, Montague semantics has largely been in the care of linguists rather than mathematicians. This meant sparse opportunity for the ideas to 'update' according to mainstream developments in mathematics.

THERE IS A NATURAL DIVISION OF MONTAGUE'S APPROACH INTO TWO STRUCTURAL COMPONENTS. First, the notion of a structure-preserving map from syntax to semantics. Second, the use of a powerful and expressive logic for semantics. We acknowledge the importance of the latter for formal semantic engineering, but here we will focus on just the former. According to Partee [], a formal semanticist, advocate, and torch-bearer for Montague, the chief interest of Montague's approach (as far as his contemporary *linguists* were concerned) lay in the following ideas:

1. Take truth conditions to be the essential data of semantics.

2. (a) Use lambdas to emulate the structure of syntax...

(b) ...in a typed system of intensional predicate logic, such that composition is function application.

I have split the second point to highlight the role of lambdas. This element was the crux of the Montagovian revolution: according to Janssen in a personal communication with Partee from 1994, lambdas were "...the feature that made compositionality possible at all."

In Section 1 of *Universal Grammar*, Montague's first paragraph establishes common notions of relation and function – the latter he calls *operation*, to distinguish the *n*-ary case from the unary case which he calls *function*. This is all done with ordinals indexing lists of elements of an arbitrary but fixed set A, which leads later on to nested indices and redundancy by repeated mention of A. We will try to avoid these issues going forward by eliding some data where there is no confusion, following common modern practice.

Next, Montague introduces his notion of *algebra* and *homomorphism*. He separates the data of the carrier set and the generators from the *polynomial operations* that generate the term algebra.

Definition 1.4.2 is equivalent to asking for all projections. Definitions 1.4.2 and 1.4.4 together characterise Montagovian algebras as (concrete) clones [].

These are (concrete) clones [] and their homomorphisms. In modern terms, (abstract) clones known to be in bijection with Lawvere theories [] ——- .

1.4.3   On Syntax

In Section 2, Montague seeks to define a broad conception of 'syntax', which he terms a *disambiguated language*. This is a free clone with carrier set A, generating operations F_γ indexed by $\gamma \in \Gamma$, along with extra decorating information:

1. $(\delta \in)\Delta$ is an (indexing) set of syntactic categories (e.g. NP, V, etc.). Montague calls this the *set of category indices*. $X_\delta \subseteq A$ form the *basic expressions* of type δ in the language.

2. a set S assigns types among $\delta \in \Delta$ to the inputs and output of – not necessarily all – F_γ.

3. a special $\delta_0 \in \Delta$ is taken to be the type of declarative sentences.

This definition is already considerably progressive. Because there is no condition of disjointness upon the X_δ – a view that permits consideration of the same word playing different syntactic roles – (1) permits the same basic expression $x \in A$ to participate in multiple types $X_\delta \subseteq A$ ($\star$). (2) misses being a normal typing system on several counts. There is no condition requiring all F_γ to be typed by S, and no condition restricting each F_γ to appear at most once: this raises the possibilities that (†) some operations F go untyped, or that (‡) some are typed multiply.

Taking a disambiguated language $\mathfrak{U}$ on a carrier set A, Montague defines a *language* to be a pair $L := < \mathfrak{U}, R >$, where R is a relation from a subset of the carrier A to a set PE_L, the set of *proper expressions* of the language L. An

Definition 1.4.1 (Generating data of an Algebra). Let A be the carrier set, and F_γ be a set of functions $A^k \to A$ for some $k \in \mathbb{N}$, indexed by $\gamma \in \Gamma$. Denoted $\langle A, F_\gamma \rangle_{\gamma \in \Gamma}$

The following three data are taken to be common among Montague's algebras:

Definition 1.4.2 (Identities). A family of operations populated, for all $n, m \in \mathbb{N}$, $n \le m$, by an m-ary operation $I_{n,m}$, defined on all m-tuples as

$$I_{n,m}(a) = a_n$$

where a_n is the nth entry of the m-tuple a.

Definition 1.4.3 (Constants). For all elements of the carrier $x \in A$, and all $m \in \mathbb{N}$, a constant operation $C_{x,m}$ defined on all m-tuples a as:

$$C_{x,m}(a) = x$$

Definition 1.4.4 (Composition). Given an n-ary operation G, and n instances of m-ary operations $H_{1 \le i \le n}$, define the composite $G(H_i)_{1 \le i \le n}$ to act on m-tuples a by:

$$G(H_i)_{1 \le i \le n}(a) = G(H_i(a))_{1 \le i \le n}$$

N.B. the m-tuple a is copied n times by the composition. Writing out the right hand side more explicitly:

$$G\Big(\big(H_1(a) , H_2(a) , \dots , H_n(a) \big) \Big)$$

Definition 1.4.5 (Polynomial Operations). The polynomial operations over an algebra $\langle A, F_\gamma \rangle_{\gamma \in \Gamma}$ are defined to be smallest class K containing all $F_{\gamma \in \Gamma}$, identities, constants, closed under composition.

Definition 1.4.6 (Homomorphism of Algebras). h is a homomorphism from $\langle A, F_\gamma \rangle_{\gamma \in \Gamma}$ *into* $\langle B, G_\gamma \rangle_{\gamma \in \Delta}$ iff

1. $\Gamma = \Delta$ and $\forall \gamma$:

2.

admirable purpose of R appears to be to permit the modelling of *syntactic ambiguity*, where multiple elements of the term algebra $\mathfrak{U}$ (corresponding to syntactic derivations) are related to the same 'proper language expression'.

However, we see aspects where Montague would have benefited from a more modern mathematical perspective: it appears that his intent was to impose a system of types to constrain composition of operations, but the tools were not available for him. Montague addresses (†) obliquely, by defining ME_L to be the image in PE_L of R of just those expressions among A that are typed. Nothing appears to guard against (‡), which causes problems as Montague expresses structural constraints (in the modern view) in terms of constraints on the codomain of an interpreting functor (cf. Montague's notion of *generating* syntactic categories). One consquence, in conjunction with ($\star$), is that every multiply typed operation F induces a boolean algebra where the typings are the generators and the operations are elementwise in the inputs and output. Worse problems occur, as Montague's clone definition include all projectors, and when defined separately from the typing structure, these projectors may be typed in a way that permits operations that arbitrarily change types, which appears to defeat the purpose. We doubt these artefacts are intentional, so we will interpret Montague assuming his intent was a type-system as we would recognise one today.

By an evident extension of [Prop 3.51] to the typed case, a *disambiguated language* is a multi-sorted Lawvere theory without relations, where the sorts are generated from products of a pointed set $(\Delta, \delta_0 : 1 \to \Delta)$.

2
String Diagrams for Text

2.1   How do we communicate?

SPEAKERS AND LISTENERS UNDERSTAND ONE ANOTHER. As far as formal theories of syntax and semantics go, this is a miracle. It remains so even if we cautiously hedge this mundane statement to exclude pragmatics and context and only encompass small and boring fragments of factual language.

In short: there is a distinction between *grammars of the speaker* – which produce sentences – and *grammars of the listener* – which deduce sentences. Viewed as mathematical processes, the two kinds of grammars go in opposite directions; speaking grammars [e.g. string-rewrite systems] start with some structure, and require informational input [e.g. which rule comes next] to produce sentences; listening grammars [e.g. typelogical grammars] start with a sentence, and require informational input [e.g. grammatical typing and which proof rule to try next] to deduce a grammatical structure. Since we can understand each other, these two types of grammar must enjoy a systematic correspondence. The correspondence looks something like this:

placeholder

In this section I will elaborate on the above argument, and provide the first steps of a formal mathematical framework to present and organise the nature of this correspondence. This framework comes from the idea that speakers and listeners may use language as a vehicle to communicate thought; from this formal framework, we detect that the intermediate structure – that which is being communicated – leads us to the idea of text circuits.

The empirical observation that speakers and listeners can communicate is a trivial one; any account of language that does not take this interactive and procedural aspect into account is arguably lacking. Text circuit theory aims to provide the beginnings of an theory of language that accounts for not just this...

2.1.1   Speaking grammars, listening grammars

Here are some naïve observations on the nature of speaking and listening. Let's suppose that a speaker, Preube, wants to communicate a thought to Fondo. Just as a running example that does not affect the point, let's say we can gloss the thought as carrying the relations:

(alice, bob, flowers, like, give)

Preube and Fondo cooperate to achieve a minor miracle; Preube encodes his thoughts – a structure that doesn't look like a one-dimensional string of symbols – into a one-dimensional string of symbols, and Fondo does the reverse, turning a one-dimensional string of symbols into *a thought-structure like that of Preube's.* The two can do this for infinitely many thoughts, and new thoughts neither have encountered before.

What I mean by this is just that both Preube and Fondo agree on the structure of entities and relations up to the words for those entities and relations. For example, Preube could ask Fondo comprehension questions such as WHO GAVE WHAT? TO WHO?, and if Fondo can always correctly answer – e.g. BOB GAVE FLOWERS. TO CLAIRE. – then both Preube and Fondo agree on the relational structure of the communicated thought to the extent permitted by language. It may still be that Preube

placeholder

We assume Preube and Fondo speak the same language, so both know how words in their language correspond to putative building blocks of thoughts, and how the order of words in sentences and special grammatical words affect the (de/re)construction procedures.

placeholder

The nature of the problem can be summarised as an asymmetry of information. The speaker knows the structure of a thought and has to make choices to turn that thought into text. The listener knows only the text, and must make choices to deduce the thought behind it. By this perspective, language is a shared and cooperative strategy to solve this (de/en)coding interaction. I will now outline the constraints imposed by this interaction, and then explain how context-free grammars and typelogical grammars partially model these constraints.

SPEAKERS CHOOSE. The speaker Preube must supply decisions about phrasing a thought in the process of speaking it. At some point at the beginning of an utterance, Preube has a thought but has not yet decided how to say it. Finding a particular phrasing requires choices to be made, because there are many ways to express even simple relational thoughts. For example, the relational content of our running example might be expressed as

ALICE LIKES THE FLOWERS BOB GIVES CLAIRE.

BOB GIVES CLAIRE FLOWERS. ALICE LIKES THE FLOWERS.

THE FLOWERS TO CLAIRE THAT BOB GIVES ARE LIKED BY ALICE.

Whether those decisions are made by committee or coinflips, those decisions represent information that must be supplied to Preube in the process of speaking a thought.

placeholder

For this reason, we consider context-free-grammars (and more generally, other string-rewrite systems) to be *Grammars of the speaker*. The start symbol S is incrementally expanded and determined by rule-applications that are selected by the speaker. The important aspect here is that the speaker has an initial state of information S that requires more information as input in order to arrive at the final sentence.

LISTENERS DEDUCE. The listener Fondo must supply decisions about which words are grammatically related, and how. Like right-of-way in driving, sometimes these decisions are settled by convention, for example, subject-verb-object order in English. Sometimes sophisticated decisions need to be made that override or are orthogonal to conventions, and we give two examples.

Example 2.1.2 (Garden path sentences). So-called "garden path" sentences illustrate that listeners have to make choices to resolve lexical ambiguities. One such garden-path sentence is `The old man the boat`, where typically readers take `The old man` as a noun-phrase and `the boat` as another noun-phrase. We can sketch how the readers might think with a (failed) pregroup grammar derivation:

$$\frac{\dfrac{\texttt{the}: n \cdot n^{-1} \qquad \texttt{old}: n \cdot n^{-1}}{\texttt{the\_old}: n \cdot n^{-1}} \qquad \texttt{man}: n \qquad \dfrac{\texttt{the}: n \cdot n^{-1} \qquad \texttt{boat}: n}{\texttt{the\_boat}: n}}{\texttt{the\_old\_man}: n}$$

<div align="center">Not a sentence!</div>

So the reader has to backtrack, taking `The old` as a noun-phrase and `man` as the transitive verb. This yields a sentence as follows:

$$\frac{\dfrac{\dfrac{\texttt{the}: n \cdot n^{-1} \quad \texttt{old}: n}{\texttt{the\_old}: n} \quad \texttt{man}:^{-1} n \cdot s \cdot n^{-1}}{\texttt{the\_old\_man}: s \cdot n^{-1}} \quad \dfrac{\texttt{the}: n \cdot n^{-1} \quad \texttt{boat}: n}{\texttt{the\_old}: n}}{\texttt{the\_old\_man\_the\_boat}: s}$$

Garden-path sentences illustrate that listeners must make choices about what grammatical roles to assign words. We make these kinds of contextual decisions all the time with lexically ambiguous words or languages with many homophones; garden-path sentences are special in that they trick the default choices badly enough that the mental effort for correction is noticeable.

Example 2.1.3 (Ambiguous scoping). Consider the following sentence:

<div align="center"><code>Everyone loves someone</code></div>

The sentence is secretly (at least) two, corresponding to two possible parses. The usual reading is (glossed) $\forall x \exists y : \texttt{loves}(x, y)$. The odd reading is $\exists y \forall x : \texttt{loves}(x, y)$: a Raymond situation where there is a single person loved by everyone. We can sketch this difference formally using a simple combinatory categorial grammar.

$$\frac{\dfrac{\texttt{everyone}: (n \multimap s) \multimap s \qquad \texttt{loves}: n \multimap s \multimapinv n}{\texttt{everyone\_loves}: s \multimapinv n} \qquad \texttt{someone}: (s \multimapinv n) \multimap s}{\texttt{everyone\_loves\_someone}: s}$$

which is stronger than the former. The last sentence is awkward: unlike in the first two cases, the quantified variable in the (gloss) $\cdots \neg \exists x_{Person} \cdots$ of `God knows` is lost, and what is left is a literal reading $\cdots \neg \texttt{knows}(\texttt{God}, \cdots) \cdots$. Second, whether a sentence is grammatically acceptable may depend on whether an exclamation is involved. `God knows` and `who knows` can be shuffled into the sentence to behave as an intensifier as in:

<div align="center">Bob drinks God knows how many beers</div>

<div align="center">Bob drinks who knows how many beers</div>

But it is awkward to have:

<div align="center">Bob drank nobody knows how many beers</div>

And it is not acceptable to have:

<div align="center">Bob drank Alice knows how many beers</div>

$$\frac{\lambda(\lambda x.V(x)).\forall x : V(x)}{\texttt{everyone} : (n \multimap s) \multimap s} \quad \frac{\texttt{loves} : n \multimap s \multimap n \quad \texttt{someone} : (s \multimap n) \multimap s}{\texttt{loves\_someone} : n \multimap s}$$
$$\frac{}{\texttt{everyone\_loves\_someone} : s}$$

CCGs have functorial semantics in any cartesian closed category, such as one where morphisms are terms in the lambda calculus and composition is substitution []. So we might specify a semantics as follows:

$$\llbracket\texttt{everyone}\rrbracket = \lambda(\lambda x.V(x)).\forall x : V(x) \tag{2.1}$$

$$\llbracket\texttt{loves}\rrbracket = \lambda x\lambda y.\texttt{loves}(x, y) \tag{2.2}$$

$$\llbracket\texttt{someone}\rrbracket = \lambda(\lambda y.V(y)).\exists y : V(y) \tag{2.3}$$

Now we can plug-in these interpretations to obtain the two different meanings. We decorate with corners just to visually distinguish which bits are partial first-order logic.

$$\frac{\lambda(\lambda x.V(x)).\ulcorner\forall x : V(x)\urcorner : (n \multimap s) \multimap s \quad \lambda x\lambda y.\ulcorner\texttt{loves}(x, y)\urcorner : n \multimap s \multimap n}{\lambda y.\ulcorner\forall x : \texttt{loves}(x, y)\urcorner : s \multimap n} \quad \lambda(\lambda y.V(y)).\ulcorner\exists y : V(y)\urcorner : (s \multimap n) \multimap s$$
$$\frac{}{\ulcorner\exists y\forall x : \texttt{loves}(x, y)\urcorner : s}$$

$$\frac{\lambda(\lambda x.V(x)).\ulcorner\forall x : V(x)\urcorner : (n \multimap s) \multimap s \quad \frac{\lambda x\lambda y.\ulcorner\texttt{loves}(x, y)\urcorner : n \multimap s \multimap n \quad \lambda(\lambda y.V(y)).\ulcorner\exists y : V(y)\urcorner : (s \multimap n) \multimap s}{\lambda x.\ulcorner\exists y : \texttt{loves}(x, y)\urcorner : n \multimap s}}{\ulcorner\forall x\exists y : \texttt{loves}(x, y)\urcorner : s}$$

THE SPEAKER'S CHOICES AND THE LISTENER'S DEDUCTIONS MUST BE RELATED. Even if the pair are cooperating, so that the speaker tries to avoid ambiguity,

2.1.2 *A context free grammar to generate* `Alice sees Bob quickly run to school`

We can describe a context-free grammar with the same combinatorial rewriting data that specifies string diagrams. For this example, we take the following n-categorical signature. The generators subscripted *L* (for *label* or *leaf*) correspond to terminals of the CFG, and represent a family of generators indexed by a lexicon for the language. The generators subscripted *i* (for *introducing a type*) correspond to rewrites of the CFG.

0-cells

$\star$

1-cells

S   NP   VP   ADV   ADP

2-cells

N_L IV_L ADV_L ADP_L

SCV_i IV_i ADV_i ADP_i
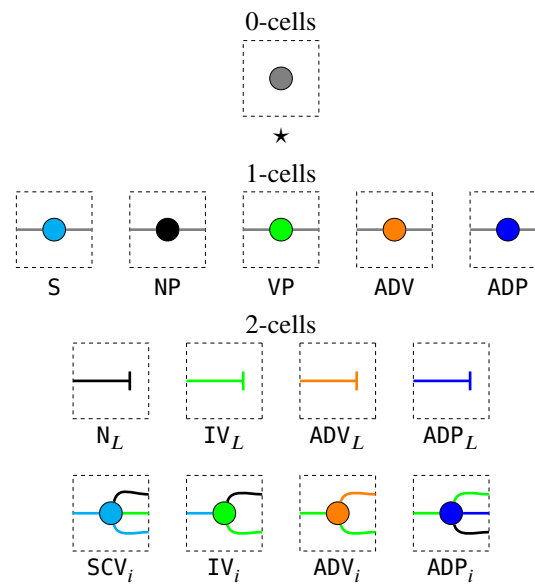
Consider the sentence `Alice sees Bob quickly run to school`, which we take to be generated by the following context-free grammar derivation, read from left-to-right. We additionally depict the breakdown of the derivation in terms of rewrites of lower dimension from our signature.
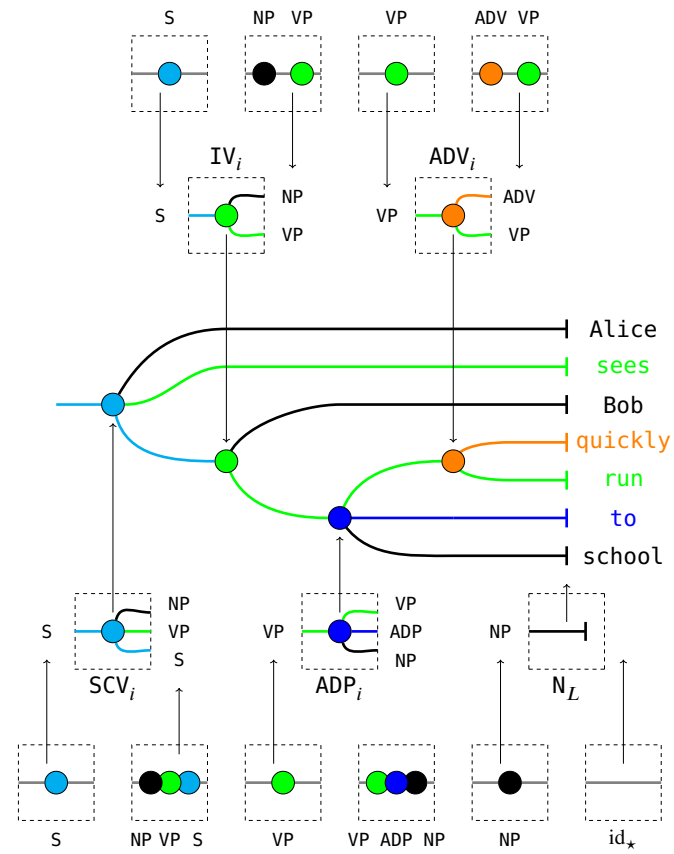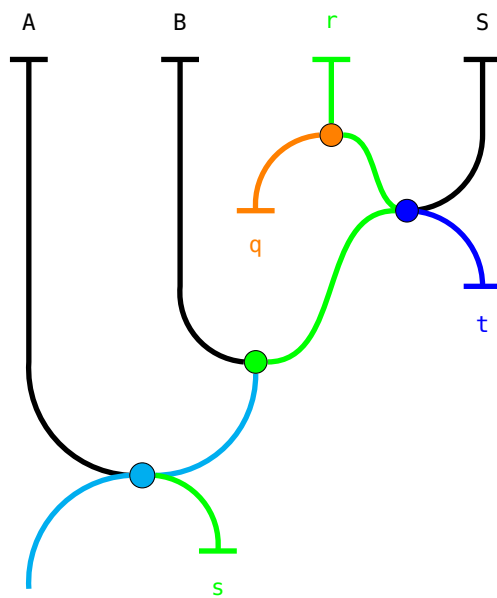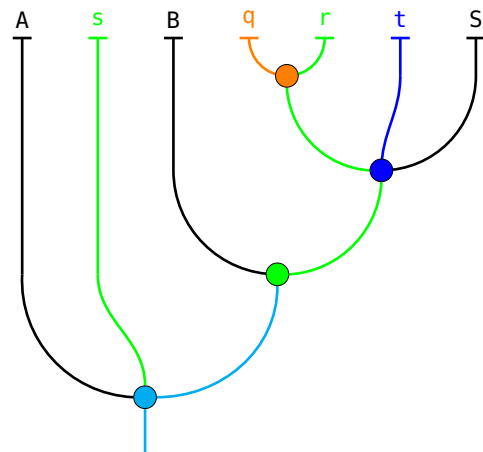
2.1.3   *Relating the generative grammar to a pregroup grammar via a discrete monoidal fibration*

We merge the monoidal functor-boxes and we slide the bottom edge down using the fibration.
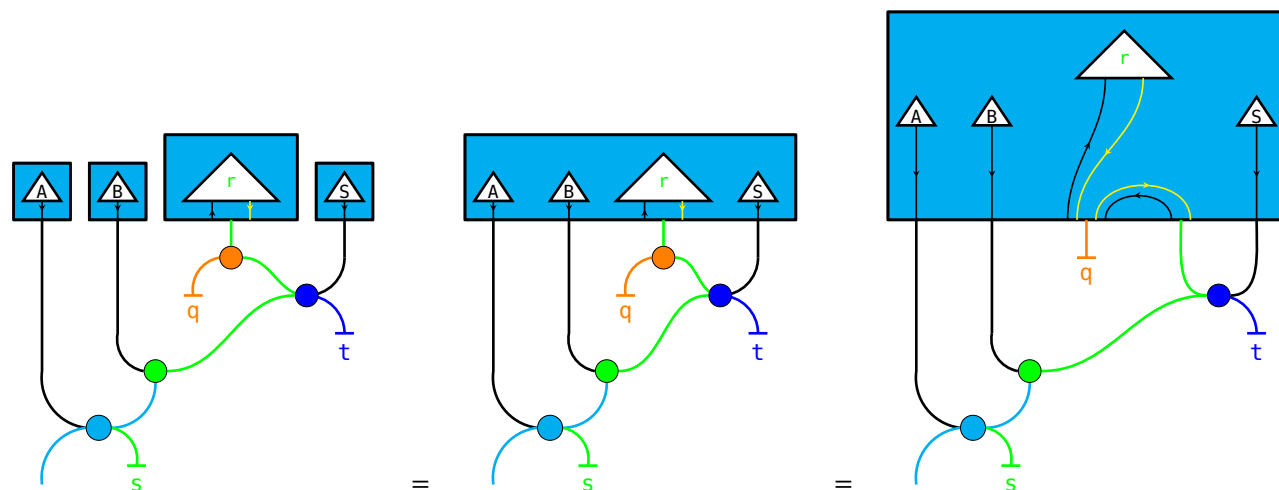
quickly could find itself modifying an intransitive (single noun) or transitive (two noun) verb. Suppose that it is the job of some process q' to handle intransitive verbs, and similarly q" to handle transitive ones. We use the functor for bookkeeping, by asking it to send both q' and q" to the dependent label q̄. Diagrammatically, this assignment is expressed by the following equations:
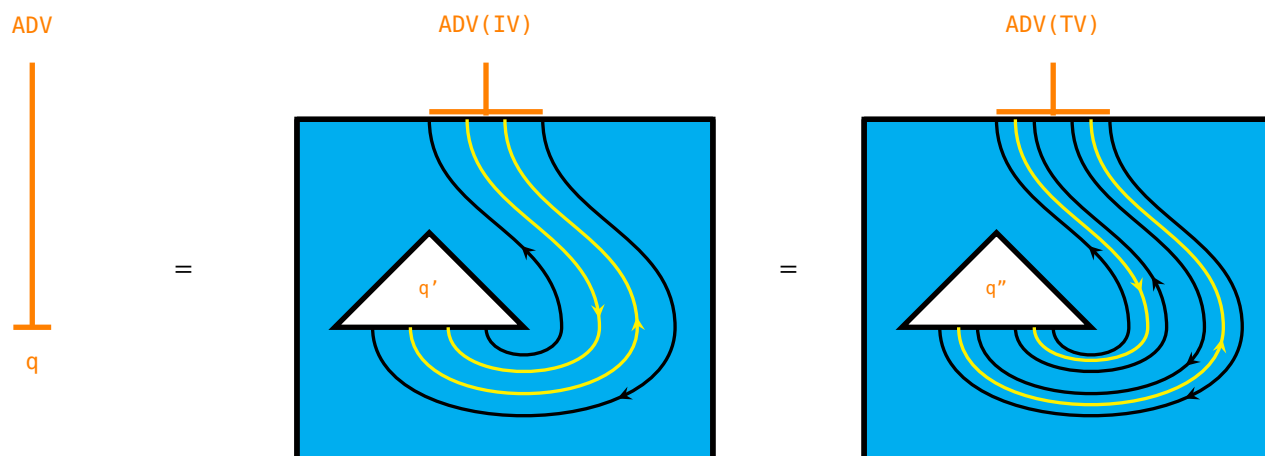
Since the functor is a monoidal discrete fibration, it introduces the appropriate choice of `quickly` when we pull the functor-box down, while leaving everything else in parallel alone.
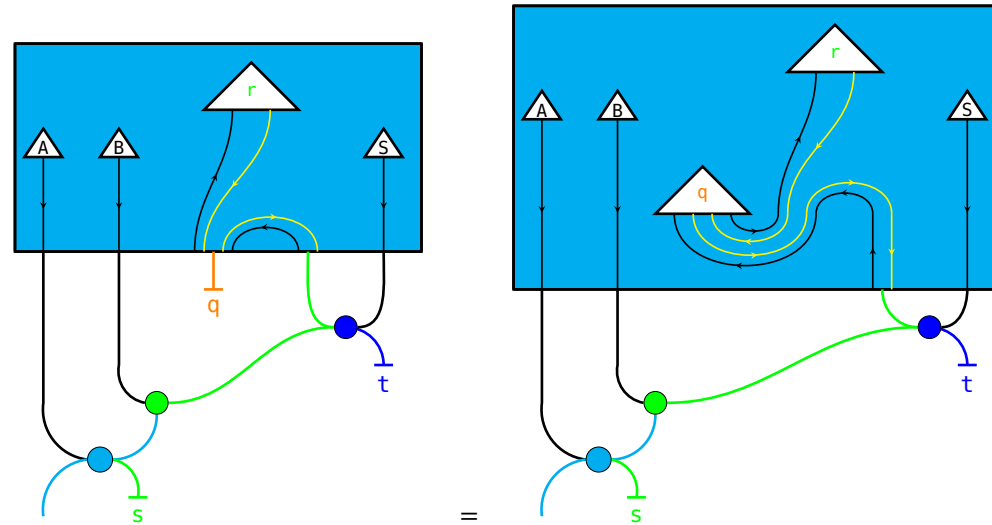
Adpositions can apply for verbs of any noun-arity. We again use the fiber of the functor for bookkeeping by asking it to send all of the following partial pregroup diagrams to the adposition generator. We consider the pregroup typing of a verb of noun-arity $k \geq 1$ to be $^{-1}n \cdot s \cdot \underbrace{n^{-1} \cdots n^{-1}}_{(k-1)}$.
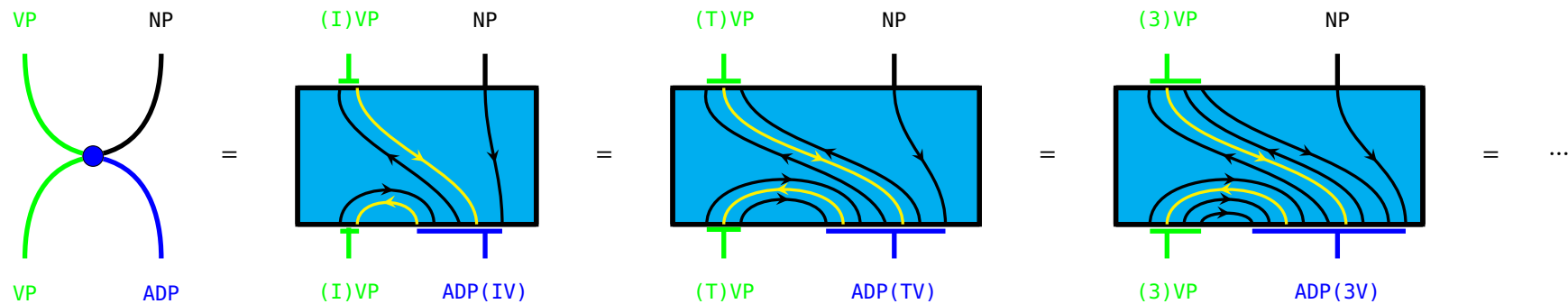
When we pull down the functor-box, the discrete fibration introduces the appropriate choice of diagram from above, corresponding to the intransitive verb case.
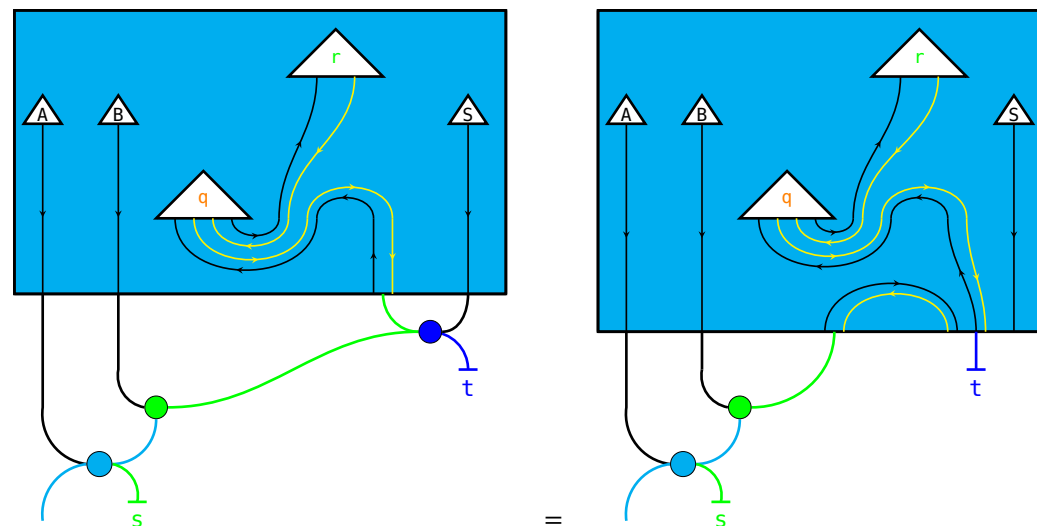
Similarly to `quickly`, we suppose we have a family of processes for the word `to`, one for each noun-arity of verb.
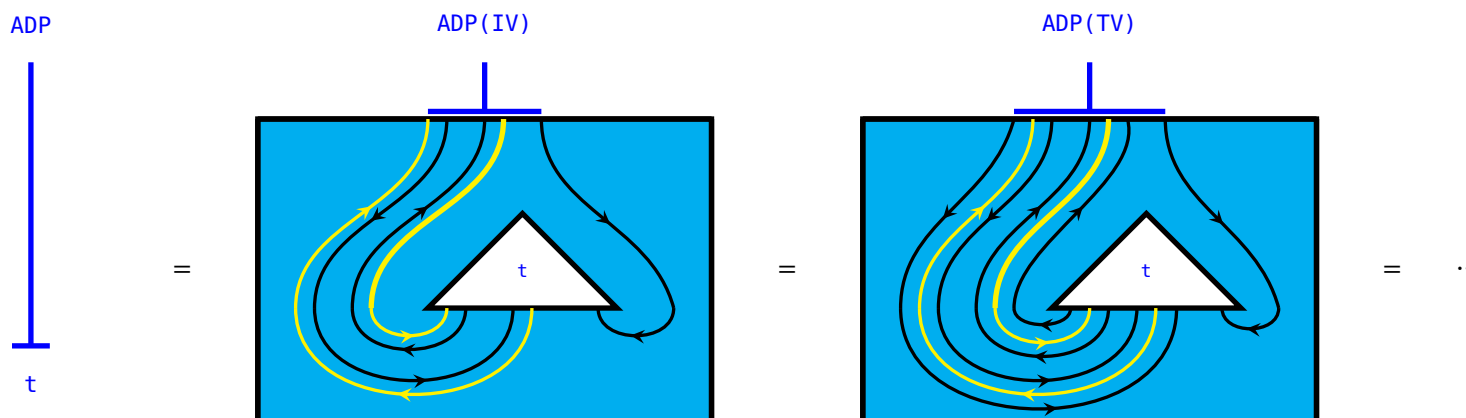
Again the discrete fibration introduces the appropriate choice of to when we pull the functor box down.

=
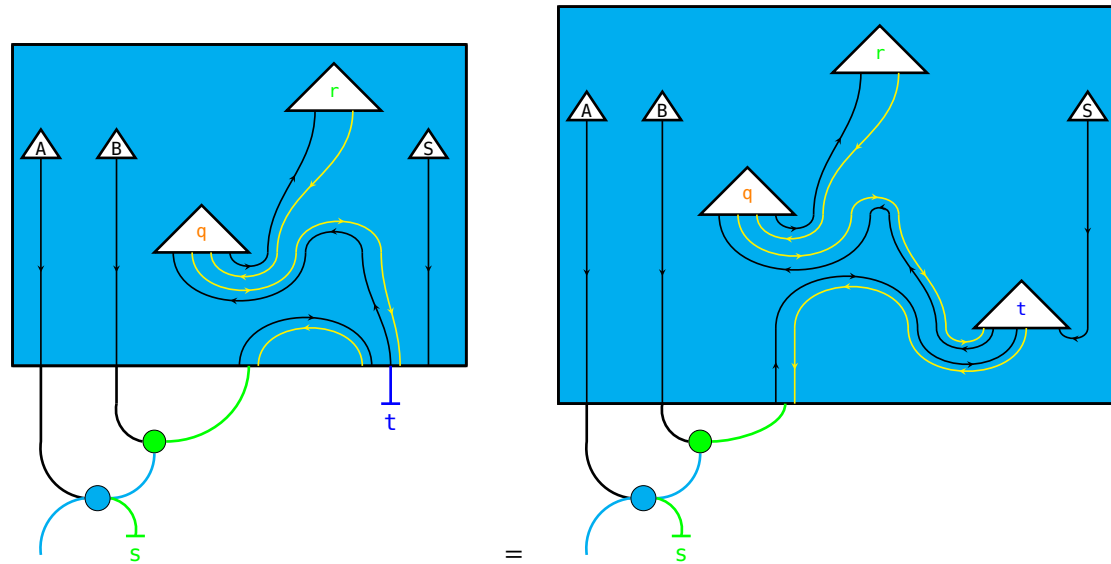
Now we visually simplify the inside of the functor-box by applying yanking equations.

=
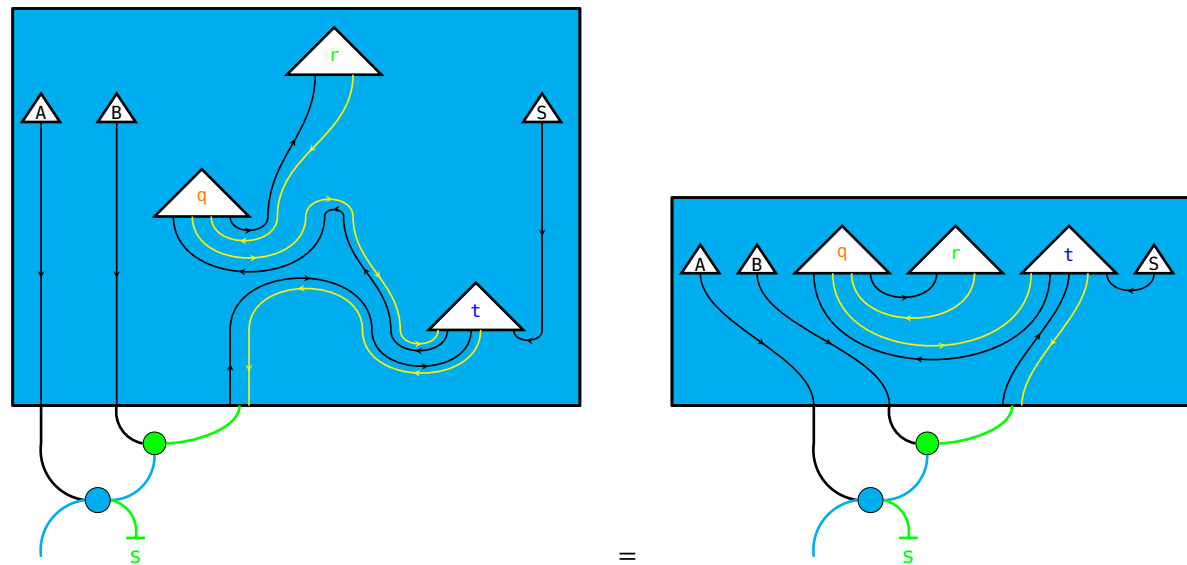
Similarly as before, we can pull the functor-box past the intransitive verb node. There is only one pregroup type $^{-1}n \cdot s$ that corresponds to the grammatical category `(I)VP`.
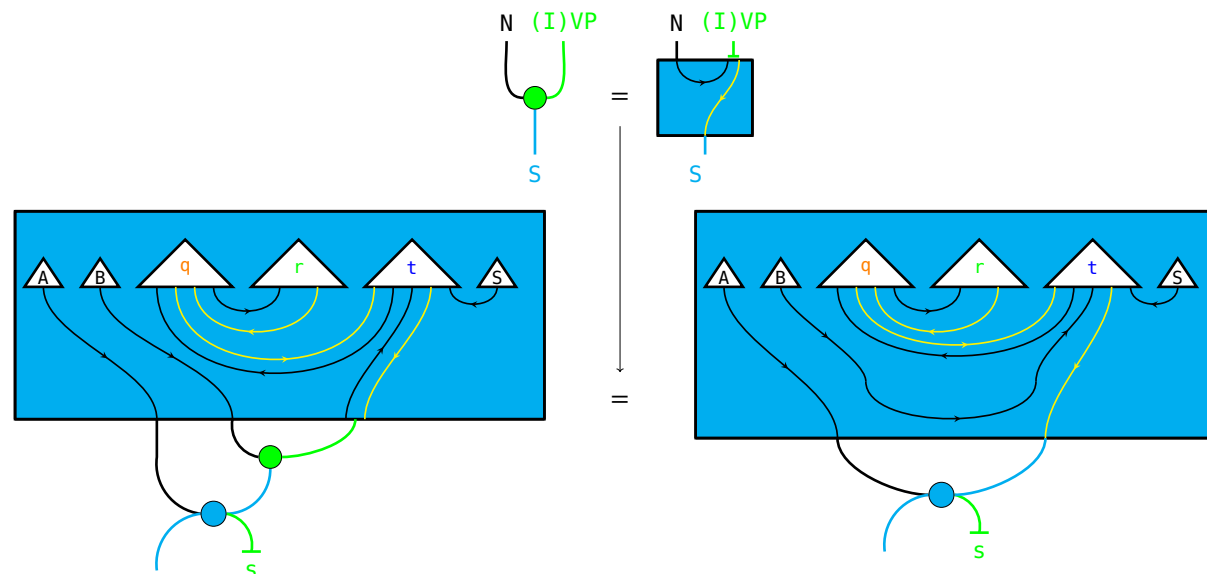
Proceeding similarly, we can pull the functor-box past the sentential-complement-verb node. There are multiple possible pregroup types for `SCV`, depending on how many noun-phrases are taken as arguments in addition to the sentence. For example, in `Alice sees [sentence]`, `sees` returns a sentence after taking a noun to the left and a sentence to the right, so it has pregroup typing $^{-1}n \cdot s \cdot s^{-1}$. On the other hand, for something like `Alice tells Bob [sentence]`, `tells` returns a sentence after taking a noun (the teller) to the left, a noun (the tellee) to the left, and a sentence (the story) to the left, so it has a pregroup typing $^{-1}n \cdot s \cdot n^{-1} \cdot s^{-1}$. These two instances of sentential-complement-verbs are introduced by different nodes. We can record both of these pregroup typings in the functor by asking for the following:
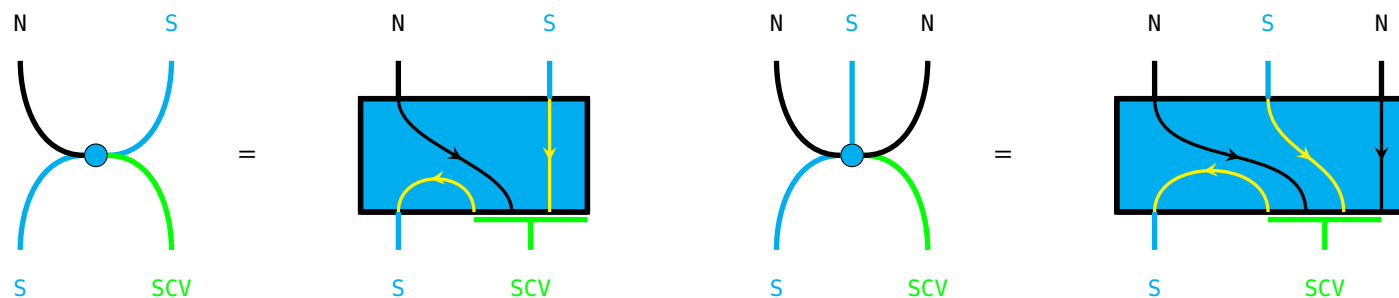
Pulling down the functor box:

$=$
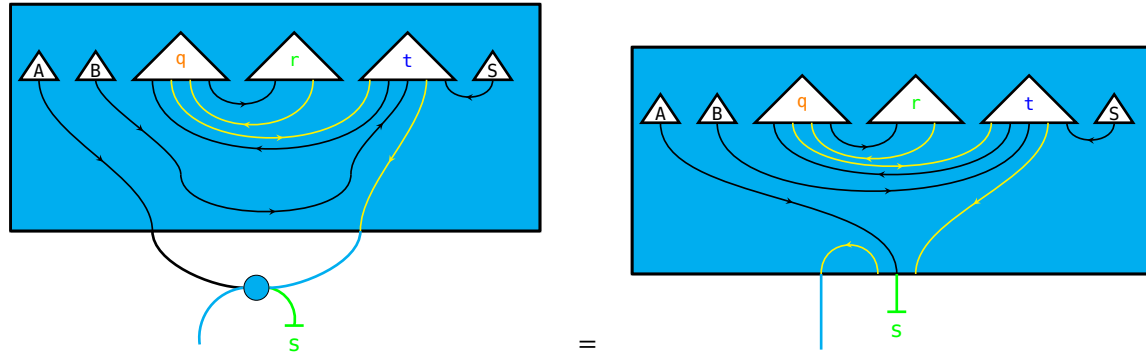
As before, we can ask the functor to send an appropriate partial pregroup diagram to the dependent label sēe.
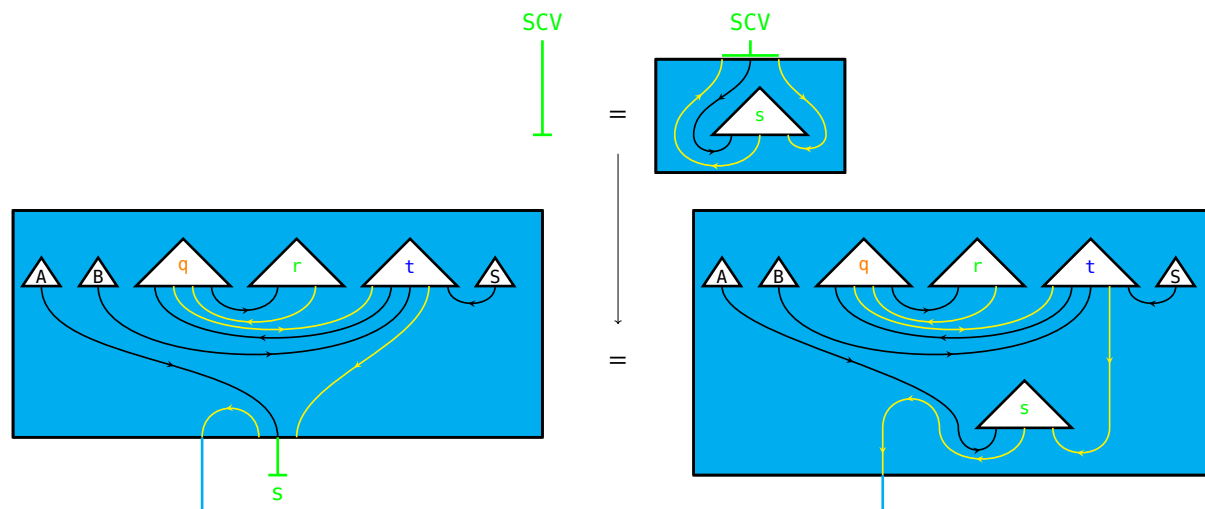
Now again we can visually simplify using the yanking equation and isotopies, which obtains a pregroup diagram.
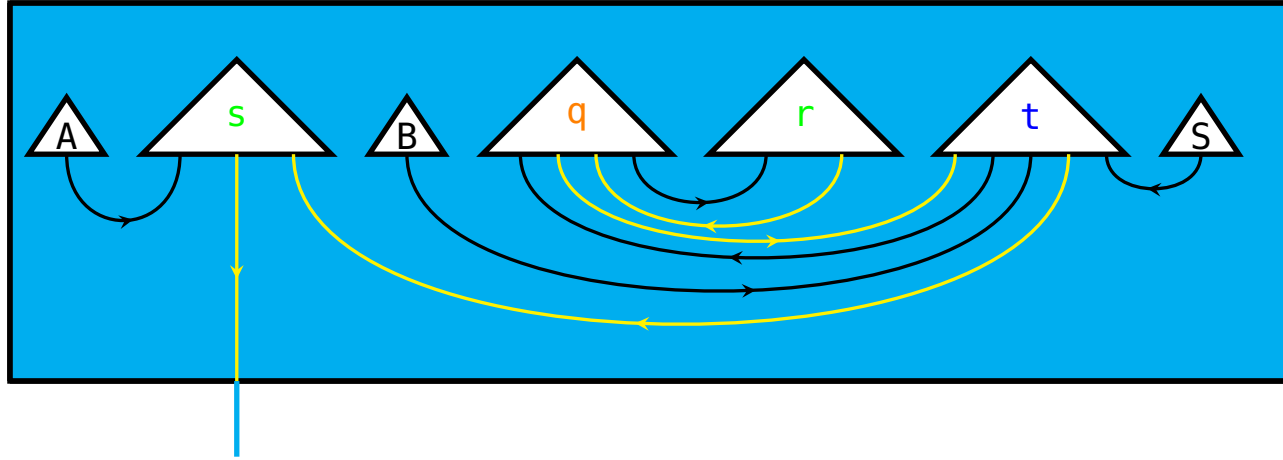
The pregroup diagram corresponds to a particular pregroup proof of the syntactic correctness of the sentence `Alice sees Bob run quickly to school.`

$$
\cfrac{A : n \qquad \cfrac{s :^{-1} n \cdot s \cdot s^{-1}}{A\_s : s \cdot s^{-1}} \qquad \cfrac{B : n \qquad \cfrac{\cfrac{\cfrac{q : (^{-1}n \cdot s) \cdot (^{-1}n \cdot s)^{-1} \qquad r :^{-1} n \cdot s}{q\_r :^{-1} n \cdot s} \qquad t :^{-1} (^{-1}n \cdot s) \cdot (^{-1}n \cdot s) \cdot n^{-1}}{q\_r\_t : (^{-1}n \cdot s) \cdot n^{-1}} \qquad S : n}{q\_r\_t\_S :^{-1} n \cdot s}}{B\_q\_r\_t\_S : s}}{A\_s\_B\_q\_r\_t\_S : s}
$$

Remark 2.1.4. Technical addendums.

The above construction only requires the source category of the functor to be rigid autonomous. Since no braidings are required, the free autonomous completion [Antonificaton] of any monoidal category may be used.

To enforce the well-definedness of the functor $\mathbf{F} : \mathcal{PG} \to \mathcal{G}$ on objects, we may consider the strictified "category of..." [ghicadiagrams]...

2.1.4 *Discrete Monoidal Fibrations*

We introduce the concept of a discrete monoidal fibration: a mathematical bookkeeping tool that relates kinds of choices speakers and listerns make when generating and parsing text respectively. We proceed diagrammatically. The first concept is that of a *monoidal functor box*.

Suppose we have a functor between monoidal categories $\mathbf{F} : \mathcal{C} \to \mathcal{D}$. Then we have the following diagrammatic representation of a morphism $\mathbf{F}A \to \mathbf{F}B$ in $\mathcal{D}$:

Scholium 2.1.5. Functor boxes are from [meillies]. Expressing the coherence conditions of monoidal functors using equations involving functor boxes as below is not new [meillies]. The idea of a functor being simultaneously monoidal and a fibration is not new [monoidalfibration]. What is new is minor: the express requirement that the lifts of the fibration satisfy interchange, which is in general not guaranteed by just having a functor be monoidal and a (even discrete) fibration [fosco].
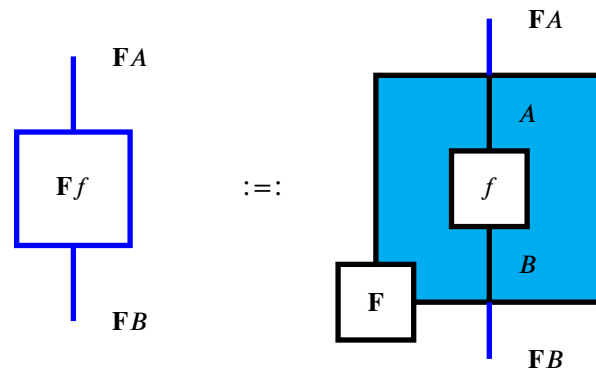
The use of a functor box is like a window from the target category $\mathcal{D}$ into the source category $\mathcal{C}$; when we know that a morphism in $\mathcal{D}$ is the image under $\mathbf{F}$ of some morphism in $\mathcal{C}$, the functor box notation is just a way of presenting all of that data at once. Since $\mathbf{F}$ is a functor, we must have that $\mathbf{F}f; \mathbf{F}g = \mathbf{F}(f; g)$. Diagrammatically this equation is represented by freely splitting and merging functor boxes vertically.
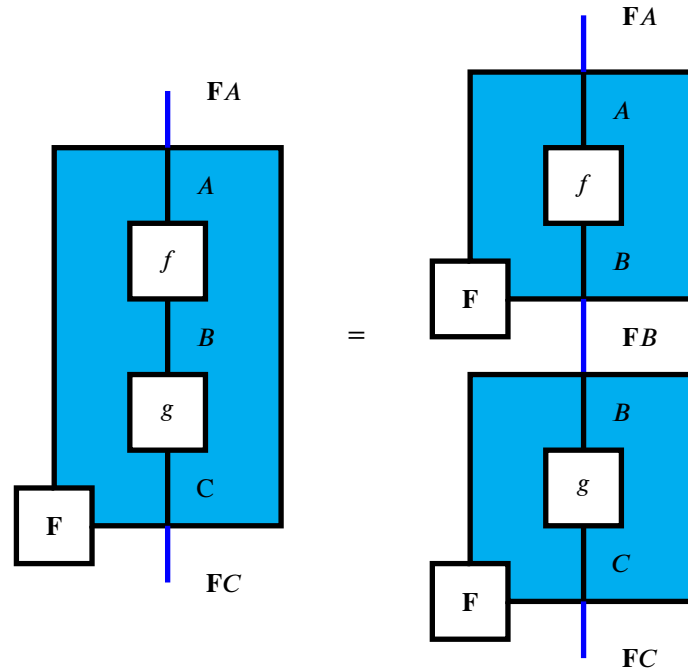
Assume that $\mathbf{F}$ is strict monoidal; without loss of generality by the strictification theorem [], this lets us gloss over the associators and unitors. For $\mathbf{F}$ to be strict monoidal, it has

to preserve monoidal units and tensor products on the nose: i.e. $\mathbf{F}I_C = I_D$ and $\mathbf{F}A \otimes_D \mathbf{F}B = \mathbf{F}(A \otimes_C B)$. Diagrammatically these structural constraints amount to the following equations:
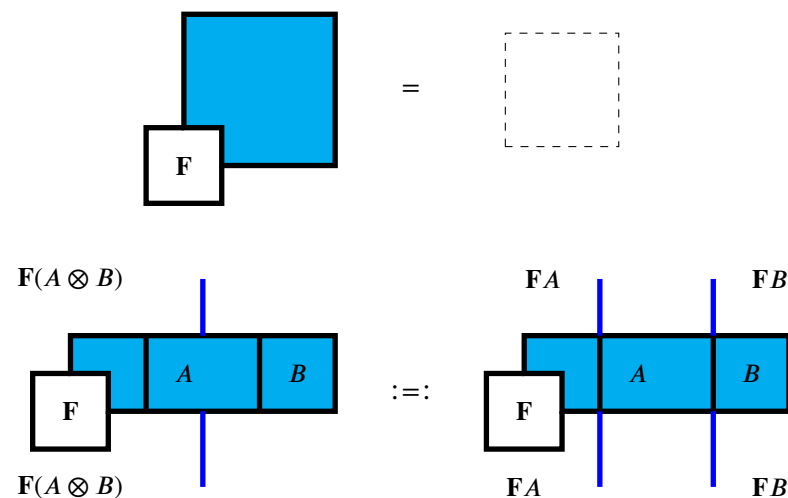
What remains is the monoidality of $\mathbf{F}$, which is the requirement $\mathbf{F}f \otimes \mathbf{F}g = \mathbf{F}(f \otimes g)$. Diagrammatically, this equation is represented by freely splitting and merging functor boxes horizontally; analogously to how splitting vertically is the functor-boxes' way of respecting sequential composition, splitting horizontally is how they respect parallel composition.
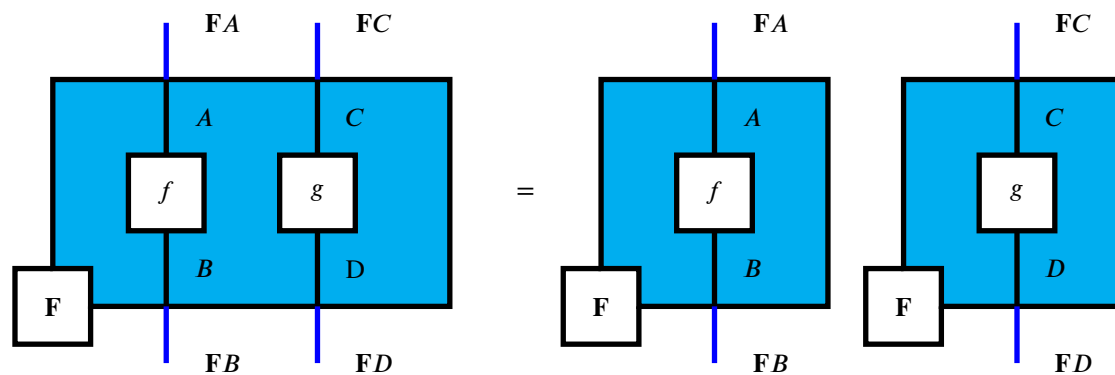
And for when we want **F** to be a (strict) symmetric monoidal functor, we are just asking that boxes and twists do not get stuck on one another.
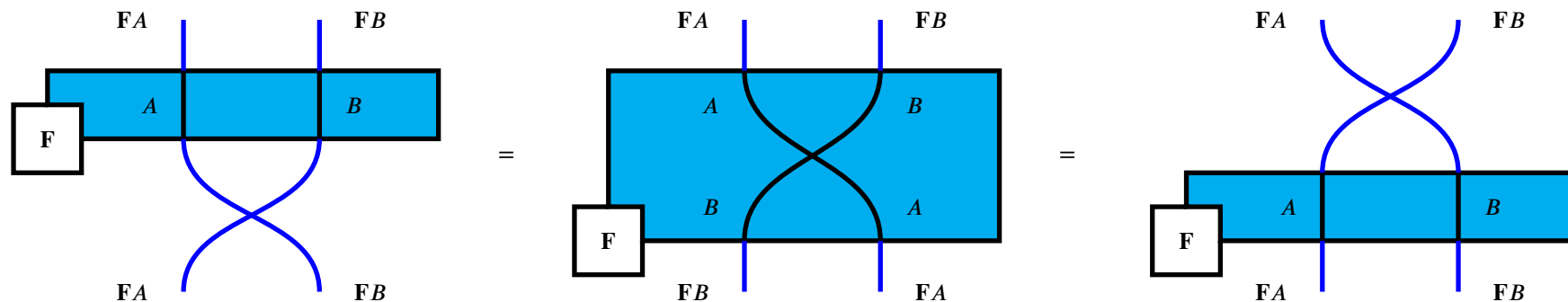
Remark 2.1.6. To motivate fibrations, first observe that by the diagrammatic equations of monoidal categories and functor boxes we have so far, we can always "slide out" the contents of a functor box out of the bottom:
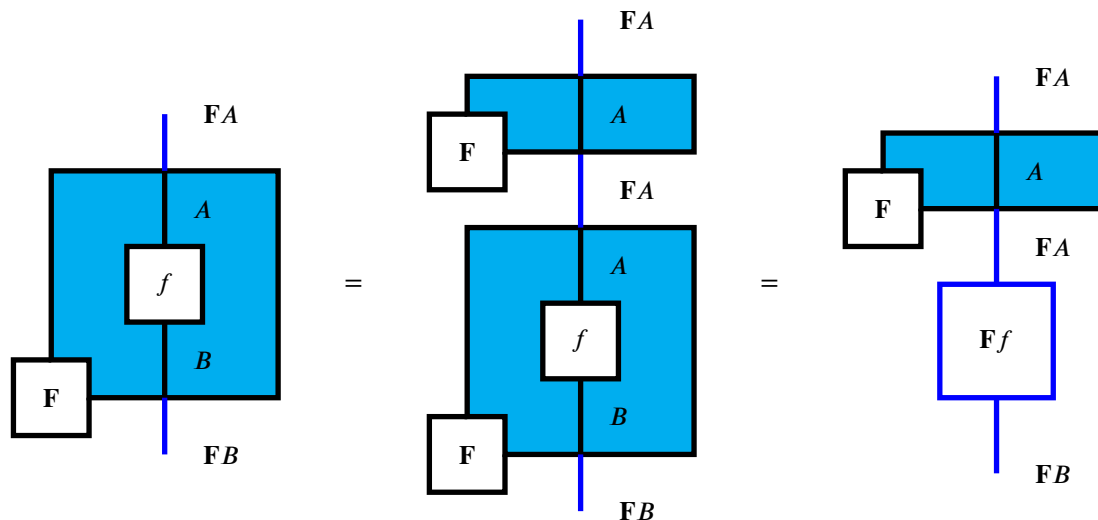
When can we do the reverse? That is, take a morphism in $\mathcal{D}$ and *slide it into* a functor box? We know that in general this is not possible, because not all morphisms in $\mathcal{D}$ may be in the image of **F**. So instead we ask "under what circumstances" can we do this for a functor **F**? The answer is when **F** is a discrete fibration.

Definition 2.1.7 (Discrete opfibration). **F** $: \mathcal{C} \to \mathcal{D}$ is a *discrete fibration* when:

for all morphisms $f : \mathbf{F}A \to B$ in $\mathcal{D}$ with domain in the image of **F**...

there exists a unique object Φ_f^A and a unique morphism $\phi_f : A \to \Phi_f^A$ in $\mathcal{C}$...

such that $f = \mathbf{F}\phi_f$.

Diagrammatically, we can present all of the above as an equation reminiscent of sliding a morphism *into* a functor box from below.

$$\forall f : \mathbf{F}A \to B \in \mathcal{D}$$

$$\exists! \varphi_f : A \to \Phi_f^A \in \mathcal{C}$$

Definition 2.1.8 (Monoidal discrete opfibration). We consider **F** to be a *(strict, symmetric) monoidal discrete opfibration* when it is a (strict, symmetric) monoidal functor, a discrete opfibration, and the following equations relating lifts to interchange hold:

Remark 2.1.9. The diagrammatic motivation for the additional coherence equations is that – if we view the lifts of opfibrations as sliding morphisms into functor boxes – we do not want the order in which sliding occurs to affect the final result. In this way, lifts behave as 'graphical primitives' in the same manner as interchange isotopies and

2.1.5   Discrete monoidal fibrations for grammatical functions

2.1.6   Extended analysis: Tree Adjoining Grammars

Here is a formal but unenlightening defintion of tree adjoining grammars, which we will convert to diagrams.

Definition 2.1.10 (Tree Adjoining Grammar: Classic Computer Science style). A **TAG** is a tuple $(\mathcal{N}, \mathcal{N}^{\downarrow}, \mathcal{N}^{\star}, \Sigma, \mathcal{I}, \mathcal{A}, \mathbf{s} \in \mathcal{N})$. These denote, respectively:

- The *non-terminals*:
  - A set of *non-terminal symbols* $\mathcal{N}$ – these stand in for grammatical types such as NP and VP.
  - A bijection $\downarrow \colon \mathcal{N} \to \mathcal{N}^{\downarrow}$ which acts as $\mathsf{X} \mapsto \mathsf{X}^{\downarrow}$. Nonterminals in $\mathcal{N}$ are sent to marked counterparts in $\mathcal{N}^{\downarrow}$, and the inverse sends marked nonterminals to their unmarked counterparts. These markings are *substitution markers*, which are used to indicate when certain leaf nodes are valid targets for a substitution operation – discussed later.
  - A bijection $\star \colon \mathcal{N} \to \mathcal{N}^{\star}$ – the same idea as above. This time to mark *foot nodes* on auxiliary trees, which is structural information used by the adjoining operation – discussed later.

- A set of *terminal symbols* Σ – these stand in for the words of the natural language being modelled.

- The *elementary trees*:
  - A set of *initial trees* $\mathcal{I}$, which satisfy the following constraints:
    * The interior nodes of an initial tree must be labelled with nonterminals from $\mathcal{N}$
    * The leaf nodes of an initial tree must be labelled from $\Sigma \cup \mathcal{N}^{\downarrow}$
  - A set of *auxiliary trees* $\mathcal{A}$, which satisfy the following constraints:
    * The interior nodes of an auxiliary tree must be labelled with nonterminals from $\mathcal{N}$
    * Exactly one leaf node of an auxiliary tree must be labelled with a foot node $\mathsf{X}^{\star} \in \mathcal{N}^{\star}$; moreover, this labelled foot node must be the marked counterpart of the root node label of the tree.
    * All other leaf nodes of an auxiliary tree are labelled from $\Sigma \cup \mathcal{N}^{\downarrow}$

There are two operations to build what are called *derived trees* from elementary and derived trees. These operations are called *substitution* and *adjoining*.

- *Substitution* replaces a substitution marked leaf node $\mathsf{X}^{\downarrow}$ in a tree α with another tree α' that has X as a root node.

- *Adjoining* takes auxiliary tree β with root and foot nodes $\mathsf{X}, \mathsf{X}^{\star}$, and a derived tree γ at an interior node X of γ. Removing the X node from γ separates it into a parent tree with an X-shaped hole for one of its leaves, and possibly multiple child trees with X-shaped holes for roots. The result of adjoining is obtained by identifying the root of β with the X-context of the parent, and making all the child trees children of βs foot node $\mathsf{X}^{\star}$.

The essence of a tree-*adjoining* grammar is as follows: whereas for a CFG one grows the tree by appending branches and leaves at the top of the tree (substitution), in a TAG one can also sprout subtrees from the middle of a branch (adjoining). Now we show that this gloss is more formal than it sounds, by the following steps. First we show that the 2-categorical data of a CFG can be transformed into 3-categorical data – which we call *Leaf-Ansatz* – which presents a rewrite system that obtains the same sentences as the CFG, by a bijective correspondence between composition of 2-cells in the CFG and constructed 3-cells in the leaf-ansatz. These 3-cells in the leaf ansatz correspond precisely to the permitted *substitutions* in a TAG. Then we show how to model *adjoining* as 3-cells. Throughout we work with a running example, the CFG grammar introduced earlier.

Construction 2.1.11 (Leaf-Ansatz of a CFG). Given a signature $\mathfrak{G}$ for a CFG, we construct a new signature $\mathfrak{G}'$ which has the same 0- and 1-cells as $\mathfrak{G}$. See the dashed magenta arrows in the schematic below. For each 1-cell wire type X of $\mathfrak{G}$, we introduce a *leaf-ansatz* 2-cell $\mathsf{X}^\downarrow$. For each leaf 2-cell X_L in $\mathfrak{G}$, we introduce a renamed copy X'_L in $\mathfrak{G}'$. Now see the solid magenta arrows in the schematic below. We construct a 3-cell in $\mathfrak{G}'$ for each 2-cell in $\mathfrak{G}$, which has the effect of systematically replacing open output wires in $\mathfrak{G}$ with leaf-ansatzes in $\mathfrak{G}'$.
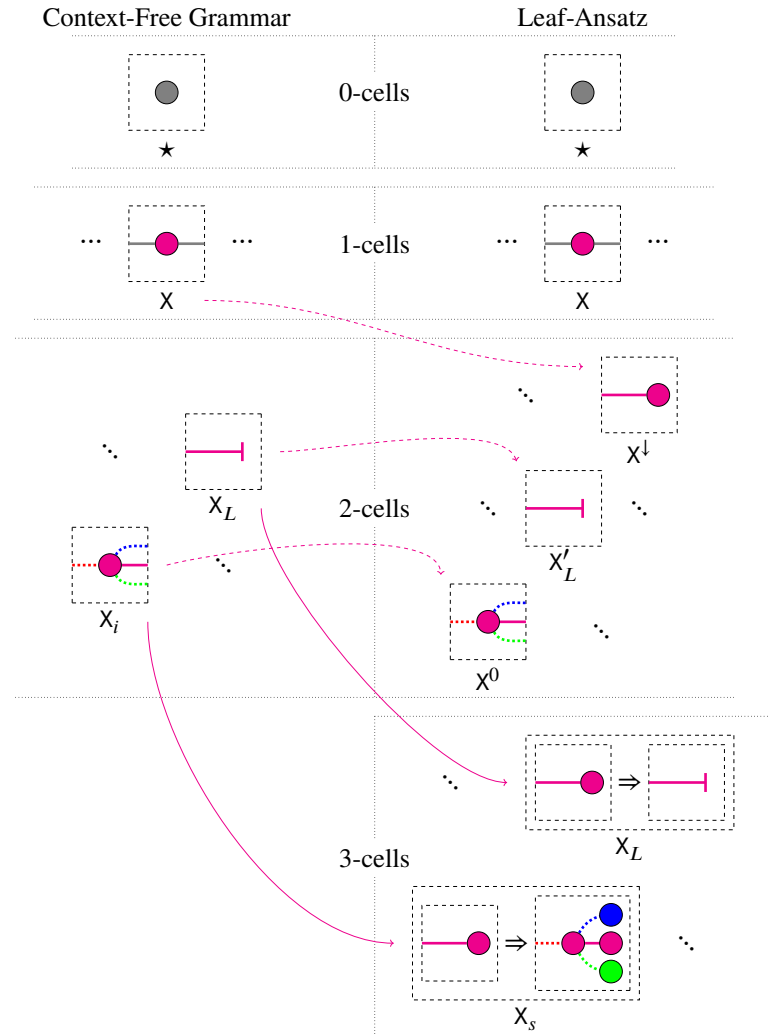
Example 2.1.12. The leaf-ansatz construction just makes formal the following observation: there are multiple equivalent ways of modelling terminal symbols in a rewrite system considered string-diagrammatically. One way (which we have already done) is to treat non-terminals as wires and terminals as effects, so that the presence of an open wire avail-

able for composition visually indicates non-terminality. Another (which is the leaf-ansatz construction) treats all symbols in a rewrite system as leaves, where bookkeeping the distinction between (non-)terminals occurs in the signature. So for a sentence like `Bob drinks`, we have the following derivations that match step for step in the two ways we have considered.
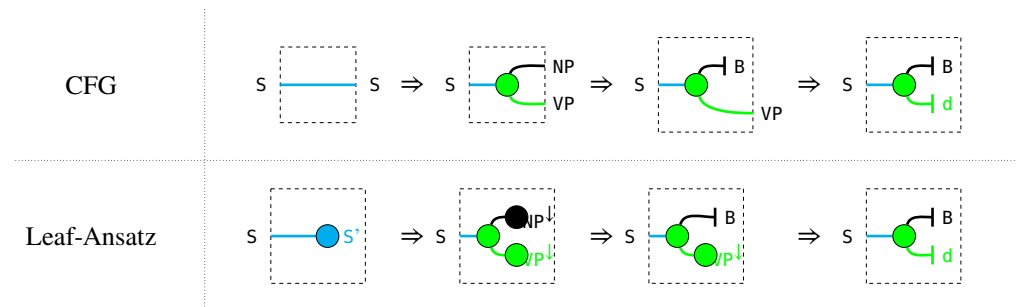
Proposition 2.1.13 (Leaf-ansatzes of CFGs are precisely TAGs with only initial trees and substitution). *Proof.* By construction. Consider a CFG given by 2-categorical signature $\mathfrak{G}$, with leaf-ansatz signature $\mathfrak{G}'$. The types X of $\mathfrak{G}$ become substitution marked symbols $\mathsf{X}^{\downarrow}$ in $\mathfrak{G}'$. The trees X_i in $\mathfrak{G}$ become initial trees X^0 in $\mathfrak{G}'$. The 3-cells X_s of $\mathfrak{G}'$ are precisely substitution operations corresponding to appending the 2-cells X_i of $\mathfrak{G}$. $\square$

Example 2.1.14 (Leaf-ansatz signature of `Alice sees Bob quickly run to school` CFG).
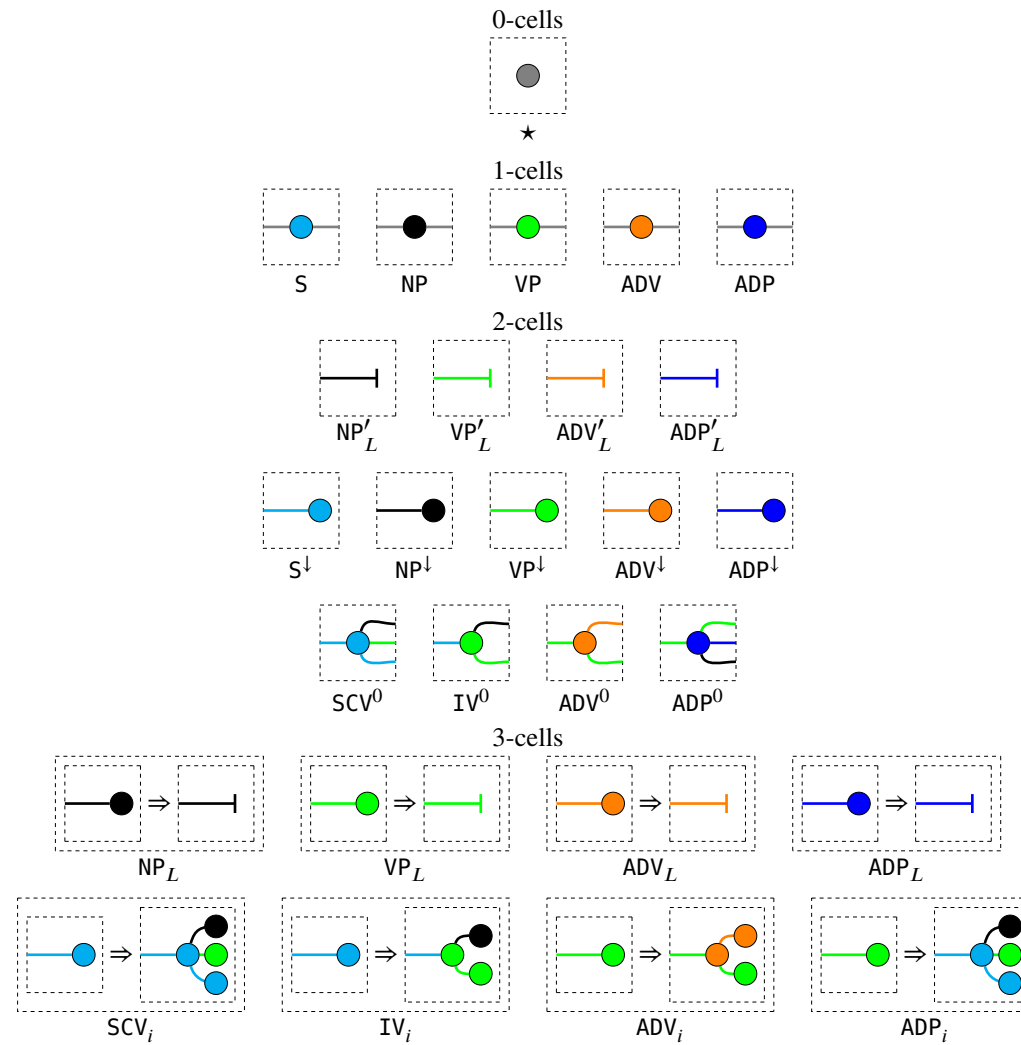
Example 2.1.15 (Adjoining is sprouting subtrees in the middle of branches). One way we might obtain the sentence `Bob runs to school` is to start from the simpler sentence `Bob runs`, and then refine the verb `runs` into `runs to school`. This refinement on part of an already completed sentence is not permitted in CFGs, since terminals can no longer be

modified. The adjoining operation of TAGs gets around this constraint by permitting rewrites in the middle of trees, as follows:
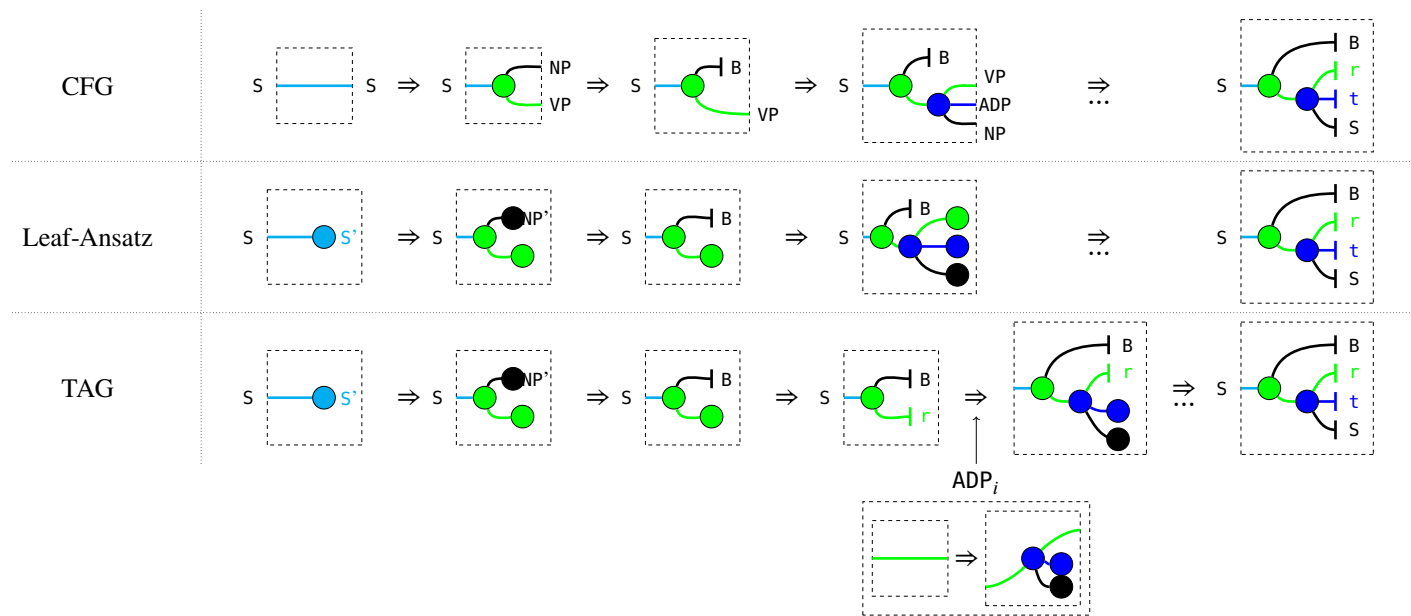
Example 2.1.16 (TAG signature of `Alice sees Bob quickly run to school`)**.** The highlighted 2-cells are auxiliary trees that replace CFG 2-cells for verbs with sentential

complement, adverbs, and adpositions. The highlighted 3-cells are the tree adjoining operations of the auxiliary trees.
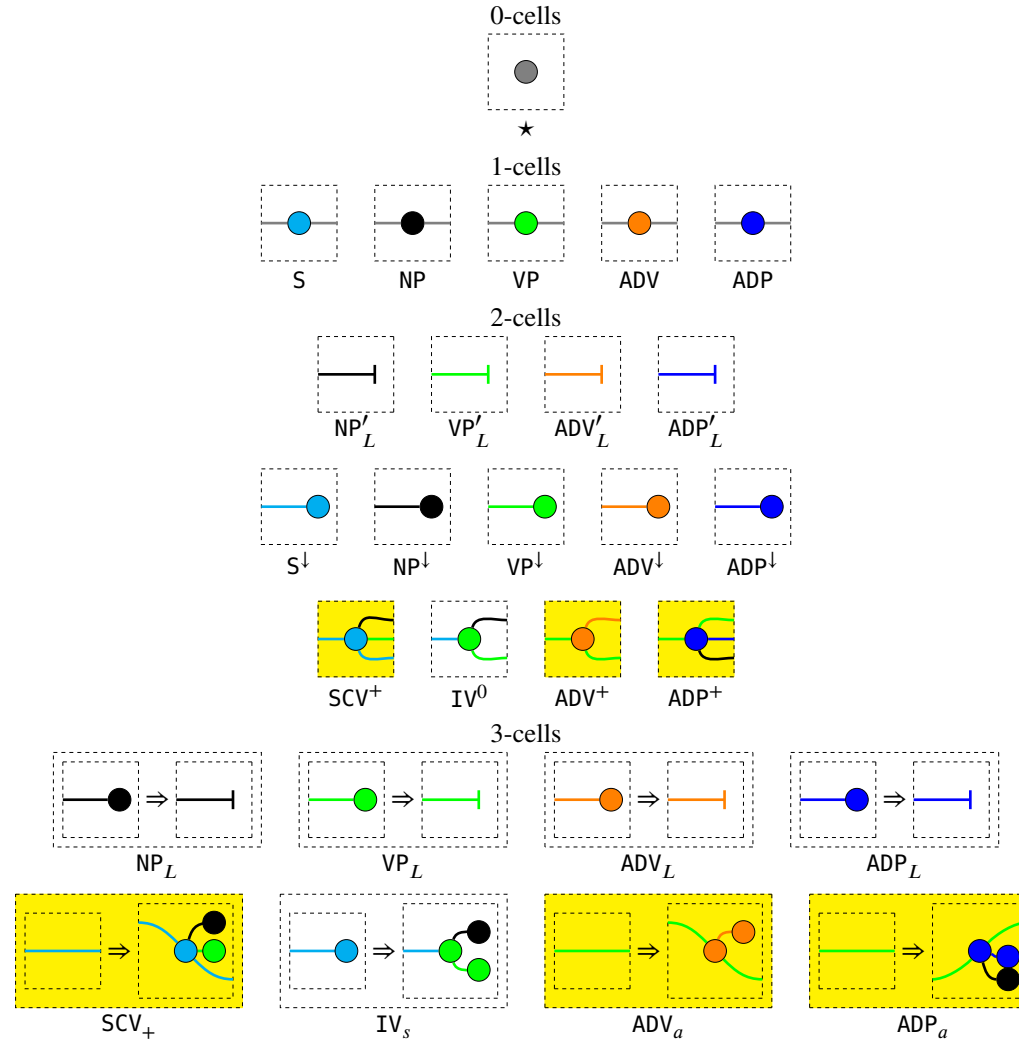
The construction yields as a corollary an alternate proof of Theorem [Joshi 6.1.1...]...

Corollary 2.1.17. For every context-free grammar $\mathfrak{G}$ there exists a tree-adjoining grammar $\mathfrak{G}'$ such that $\mathfrak{G}$ and $\mathfrak{G}'$ are strongly equivalent – both formalisms generate the same set of strings (weak equivalence) and the same abstract syntactic structures (in this case, trees) behind the strings (strong equivalence).

Proof. Proposition 2.1.13 provides one direction of both equivalences. For the other direction, we have to show that each auxiliary tree (a 2-cell) and its adjoining operation (a 3-

cell) in $\mathfrak{G}'$ corresponds to a single 2-cell tree of some CFG signature $\mathfrak{G}$, which we demonstrate by construction. See the example above; the highlighted 3-cells of $\mathfrak{G}'$ are obtained systematically from the auxiliary 2-cells as follows: the root and foot nodes $X, X^\star$ indicate which wire-type to take as the identity in the left of the 3-cell, and the right of the 3-cell is obtained by replacing all non-X open wires Y with their leaf-ansatzes $Y^\downarrow$. This establishes a correspondence between any 2-cells of $\mathfrak{G}$ considered as auxiliary trees in $\mathfrak{G}'$. $\square$
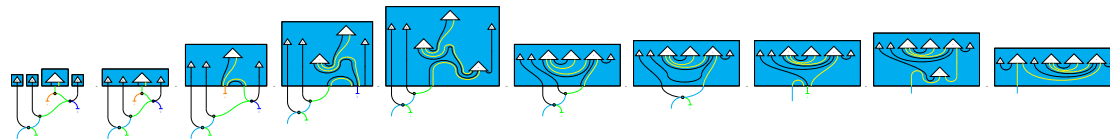
3

Continuous relations: a palette for toy models

3.1   Continuous Relations: A concept-compliant setting for text circuits

In this chapter, we introduce *continuous relations*, which are a naïve extension of the category **Top** of topological spaces and continuous functions towards continuous relations. We choose this category (as opposed to plain **Rel**, the category of sets and relations) because it satisfies several requirements arrived at by introspection of some of the demands of modelling language. These justifications involve basic reflections upon language use, and the consequent mathematical constraints those affordances impose on any interpretation of text circuits in a symmetric monoidal category; A priori it could well be that there is no non-trivial process theory that satisfies these constraints, so the onus is on us to show that there exists such a process theory. I outline these justifications – mostly intended for readers interested in how any of this relates to the cognitive aspect of text – after this subsection. The justifications can also be skipped and optionally revisited after the reader is more familiar with what **TopRel** is.

Second, we introduce **TopRel** diagrammatically. In the appendix for this chapter we do the bookwork demonstrating that it is a symmetric monoidal category, and we relate it to the well studied categories **Rel** and **Loc**. To the best of my knowledge, the study of this category is a novel contribution, for reasons I list prefacing the bookwork.

Third, once we have defined a stage to perform calculations in **TopRel**, our aim is to introduce some actors. We seek to make formal the kinds of informal schemata we might doodle on paper to animate various processes occurring in space. One good reason for doing this is to establish formal foundations for the semantics of metaphor, some of the most commonly used of which involve spatial processes in a way that is fundamentally topological []. For example, in the *conduit metaphor* [], `words` are considered *containers* for `ideas`, and `communication` is considered a *conduit* along which those containers are sent.
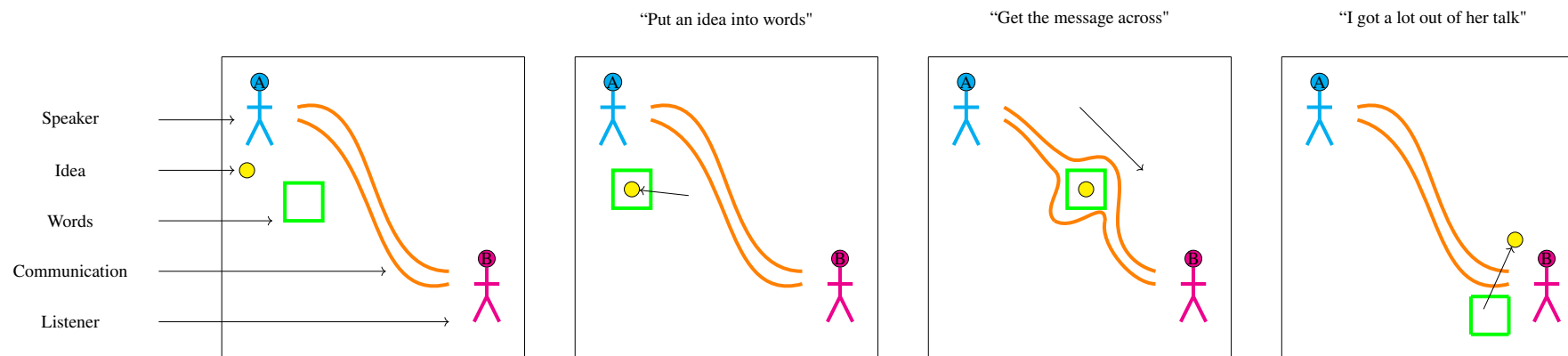
If you are already happy to treat such doodles as formal, then I think you're alright, you can skip the rest of this chapter. If you are a category theorist or just curious, hello, how are you, please do write me to share your thoughts if you care to, and please skip the rest of this paragraph. If you are still reading, I assume you are some kind of smelly epistemic-paranoiac Bourbaki-thrall sets-and-lambdas math-phallus-worshipping truth-condition-blinded symbol-pusher who takes things too seriously. I hope you choke on the math. I will begin the intimidation immediately.

We provide a generalisation (Definition 3.5.5) of special commutative frobenius algebras in **TopRel** that cohere with idempotents in the category. The relation of (†)-special commutative algebras in **FdHilb** to model observables in quantum mechanics is well-studied [], as is the role of idempotents in generalisations of quantum logic to arbitrary categories [], therefore this generalisation may be viewed as a unification of these ideas to define doodles in **TopRel**. N.B. we are not quite taking the Karoubi envelope of **TopRel**, as

we are restricting the idempotents we wish to consider to only those that also behave appropriately as observables. The reason for this restriction lies in Theorem 3.5.8 which provides a diagrammatic characterisation result that allows us to precisely identify when any idempotent in the category splits though a discrete topology. The discrete topology thus acts as a set of labels, where the (pre)images of each element under section and retract behave as the kind of shapes we wish to consider. As an interlude, we demonstrate how we may construct families of such idempotent-coherent special commutative frobenius algebras on $\mathbb{R}^2$ to provide a monoidal generalisation (c.f. the monoidal computer framework in []) of **FinRel** equipped with a Turing object [], thus satisfying Justification **??**. Then we proceed to define configuration spaces of collections of shapes up to rigid displacement, and we develop a relational analogue of homotopy to model motions as paths in configuration space, along with appropriate extensions to accommodate nonrigid phenomena. If you are still reading and not already a category theorist nor just curious, I will drop the jargon now, because you are probably either questioning or deeply committed to your false ideals, both of which I respect, but just to spite the latter, I will proceed to do everything diagrammatically as much as possible.

3.1.1   *Why not use something already out there?*

JUSTIFICATION 1: WE WANT A SYMMETRIC MONOIDAL CATEGORY. We want to work process-theoretically as much as possible, because, as we have seen, this potentially allows whatever we construct in this setting to generalise to other process theories. Also, we want an excuse to build up and play with a symmetric monoidal category from scratch, just to see what formal work is involved in doing so.

JUSTIFICATION 2: WE WANT TO MODEL CONCEPTS. Second, we have some cognitive considerations: how do we move between concepts – however they are represented – and symbolic representation and manipulation? Here we sidestep the debate around what concepts actually *are*, aligning ourselves close to Gärdenfors: we assume that there are *conceptual spaces* that organise concepts of similar domain – such as colour, taste, motion – and that regions of these spaces correspond to concepts. Gärdenfors' stance is backed by empirical data [], but even if he is wrong, he is at least interestingly so for our purposes. As an example, the classic example of colourspace is also one of the best studied and implemented: there are many different embeddings of the space of visible colours in Euclidean space []. In this setting we can mathematically model the action of categorising a particular point in colour space as `blue` by checking to see whether that point falls within the region in colourspace that the symbol `blue` is associated with. So we find that this view is conducive to modelling concepts as spatial entities – a very permissive and expressive framework, which will allow us to calculate interesting things – whilst also having the ability to handle them using symbolic labels. The question remains: how do we model this association between space and sets of labels? This leads us to the following consideration: In a category for concept-compliant text-circuits $\mathcal{T}$, any conceptual space object Γ should possess a split idempotent through a set of concept-labels, thus encoding the association between concepts-qua-spaces and concepts-qua-symbols.

A further complication arises. Once we have a stock of symbols referring to entities in space, we can start talking about pairs of entities (e.g. `red or blue`), subsets of sets of entities (e.g. `autumnal colours`), arbitrary relations between the set of entities in one space and entities of another (e.g. how the colour of a banana relates to its probable textures and tastes). Given enough time and patience, we can linguistically construct – at least – any finite relation between finite sets. As we will elaborate in Section 4, the real challenge is that every time we define a new concept in this way, we can again treat it as a symbolic concept, that is, label it with a noun. Since nouns are first-class citizens that travel along wires in our framework, we are asking that any putative process theory that satisfies these considerations about concepts has to have some object, some wire Ξ for nouns, such that all finite relations fit into it. This leads us to the following consideration: A category for concept-compliant text-circuits $\mathcal{T}$ ought to have an object Ξ such that all of **FinRel** can be encoded within Ξ somehow. But we already know how to encode using split idempotents, so we can translate the two considerations of the last two paragraphs into one requirement. In prose; the first item gives us a method within the category $\mathcal{T}$ to associate concepts-qua-spaces to concepts-qua-symbols; the second item gives us a noun-wire in $\mathcal{T}$

that permits us to encode and manipulate concepts however we would like to treat finite relations.

Requirement 3.1.1. We call a text circuit category $\mathcal{T}$ *concept-compliant* if:

1. Any wire Γ used to model a conceptual space possesses a split idempotent through a set of concept-symbols $\mathfrak{L}$

2. Anything one can do with sets of concept-symbols in **FinRel** must be doable in $\mathcal{T}$; in particular, there exists a noun-wire Ξ such that **FinRel** embeds as a category into the sub-category of $\mathcal{T}$ generated by the split idempotents on Ξ

JUSTIFICATION 3: WE THINK TOPOLOGY IS A GOOD SETTING TO MODEL CONCEPTS SPATIALLY.

Where we differ from Gärdenfors is that we only ask for topological spaces, rather than his stronger requirements for metric spaces and convex concepts, so we are following the spirit but not the letter. There are several reasons for this choice. First, topology is a primitive mathematical framework for space; all metric spaces are topological spaces, but not vice versa, so this is a conservative generalisation of Gärdenfors. Second, there are technical reasons that **TopRel** is desirable, For example, we can process-theoretically characterise continuous maps from the unit interval fairly easily to model things going on in space and time; without topology around, for instance in the setting of just **Rel**, I do not know if it is even possible to pick out the continuous maps from all the others purely process-theoretically. Third and perhaps most interestingly, there are also good reasons to think that this is the right way to think about conceptual spaces. We can view topology as a framework for conceptual spaces where we consider the open sets of a topology to be the concepts. Éscardo provides a the following correspondence [], which we extend with "Point" and "Subset", and an additional column for interpretation as conceptual space:

Topology	Type Theory	Conceptual Spaces
Space	Type	Conceptual space
Point	Element of set	Copyable instance
Subset	Subset	Instance
Open set	Semi-decidable set	Concept
Closed set	Set with semi-decidable complement	-
Clopen set	Decidable set	A concept the negation of which is also a concept
Discrete topology	Type with decidable equality	A conceptual space where any collection of instances forms a concept
Haussdorf topology	Type with semi-decidable inequality of elements	A conceptual space where any pair of distinct instances can be described as belonging to two disjoint concepts
Compact set	Exhaustively searchable set, in a finite number of steps	A conceptual space $\mathfrak{C}$ such that for any joint concept R on $\mathfrak{C} \times \mathfrak{D}$, $\forall c_{\in \mathfrak{C}} R(c, -)$ is a concept in $\mathfrak{D}$

TopRel reflects the above correspondences diagrammatically. Open sets are precisely tests, and it is always possible to construct finite intersections of opens using copy-relations. Modelling concepts as open sets or tests aligns them with semi-decidability. Given any state-instance, we can test whether it overlaps with a concept graphically: success returns a unit scalar, failure returns the zero scalar. Moreover, another independent diagrammatic calculus for concepts by Tull [] agrees with ours; concepts are effects, copy maps take intersections of concepts, and so on. Tull's diagrams are interpreted in **Stoch**, the category of stochastic processes, and as a whole his design philosophy closely adheres to Gärdenfors. All this is to suggest that if you take Gärdenfors seriously, then there is something worth taking seriously about modelling concepts as effects in a monoidal category with copy-maps. However, taking diagrammatic conceptual spaces seriously also yields a no-go result for topological spaces – which includes Gärdenfors' metric spaces with convex concepts: you can only do first-order logic with equality on your concepts if your base space is discrete and finite. We explain this below.

The correspondence between spaces and type-theory extends to conceptual spaces as follows: "niceness conditions on your conceptual space correspond to the ability to form

new concepts using logical operations." For example, this means that if we denote colourspace with $\mathfrak{C}$, we can only construct a concept `different colours` on $\mathfrak{C} \times \mathfrak{C}$ if we model $\mathfrak{C}$ using a Hausdorff space, such as Euclidean space. If we want to model `not X` as the complement of a colour `X` in colourspace, asking that `not X` also be a concept requires $\mathfrak{C}$ be locally indiscrete – i.e. every open set is also closed; Euclidean space is not locally indiscrete, so we cannot use set-complement as negation, we have to use something else, like the interior of the complement. If we want to have access to universal quantifiers in colourspace, so that we can sensibly construct concepts such as `the taste that apples of all colours have in common`, then we require $\mathfrak{C}$ to be compact, which Euclidean space is not, but bounded Euclidean space is. Here then is the conflict: if we take modelling concepts with spaces seriously, and we also care to do logic with concepts, there are tradeoffs to be made. In order to take equality as a concept `same colour` on $\mathfrak{C} \times \mathfrak{C}$ requires that $\mathfrak{C}$ have discrete topology, but discrete topologies on infinite sets are not compact, so you cannot also have universal quantification at the same time. Conversely, if you want universal quantification on colourspace, then you can at best have approximate equality on colours as a concept, never exact. Now we can summarise this tension. All topological spaces, including Gärdenfors conceptual spaces with metrics and convex concepts, start off with regular logic – $\exists, \wedge, \top$ – for free []. In order to obtain first-order logic with equality on the points of the space, the underlying space must be compact (to support universal quantification) and discrete (to support equality of points), and the latter condition implies locally indiscrete (to support negation). Only finite sets with the discrete topology are compact; you can only do first-order logic with equality on your concepts if your base space is discrete and finite.

This no-go just provides another perspective of the ancient observation that logical thought really does seem symbolic. It is also important to note that the essential idea of conceptual spaces is to circumvent this no-go; in our language, asking for split idempotents through (finite) discrete topologies is precisely what allows logical constructions to work on spatial representations of concepts, when those split idempotents exist. What this no-go does mean is that nobody has to waste their time looking for *precise* unifications of conceptual spaces and first-order logic-with-equality where every logical operator is viewed as a space-preserving transformation that sends concepts to concepts and behaves properly on the space of points; there is only "good enough".

SUMMARY OF JUSTIFICATIONS. We want a symmetric monoidal category to keep the prospect of general, process-theoretic reasoning. We want the objects of this category to be topological spaces because we want to model conceptual spaces and calculate interesting things. While the category **Top** of topological spaces and continuous functions is already symmetric monoidal with respect to categorical product, for linguistic and concept-related considerations we want finite relations to come into play diagrammatically, and because **Top** is cartesian monoidal it doesn't work for our purposes. So we construct **TopRel**.

3.2   Continuous Relations

TO THE BEST OF MY KNOWLEDGE, THE STUDY OF **TOPREL** IS A NOVEL CONTRIBUTION. I VENTURE
TWO POTENTIAL REASONS.

FIRST, IT IS BECAUSE AND NOT DESPITE OF THE NAÏVITY OF THE CONSTRUCTION. Usually, the relation-
ship between **Rel** and **Set** is often understood in sophisticated general methods which are inappropriate in different
ways. I have tried applying Kliesli machinery which generalises to "relationification" of arbitrary categories via ap-
propriate analogs of the powerset monad to relate **Top** and **TopRel**, but it is not evident to me whether there is such
a monad. The view of relations as spans of maps in the base category should work, since **Top** has pullbacks, but this
makes calculation difficult and especially cumbersome when monoidal structure is involved. The naïve approach
I take is to observe that the preimages of functions are precisely relational converses when functions are viewed
as relations, so the preimage-preserves-opens condition that defines continuous functions directly translates to the
relational case.

SECOND, THE RELATIONAL NATURE OF **TOPREL** MEANS THAT THE CATEGORY HAS POOR EXACTNESS
PROPERTIES. Even if the sophisticated machinery mentioned in the first reason do manage to work, relational
variants of **Top** are poor candidates for any kind of serious mathematics because they lack many limits and colimits.
Since we take an entirely "monoidal" approach – a relative newcomer in terms of mathematical technique – we are
able to find and make use of the rich structure of **TopRel** with a different toolkit.

In the end, we want to formalise doodles, so perhaps there is some virtue in proceeding by elementary means.

Recall that functions are relations, and the inverse image used in the definition of continuous maps is equivalent
to the relational converse when functions are viewed as relations. So we can naïvely extend the notion of continuous
maps to continuous relations between topological spaces.

Definition 3.2.4 (Continuous Relation). A continuous relation $R : (X, \tau) \to (Y, \sigma)$ is a relation $R : X \to Y$ such
that

$$U \in \sigma \Rightarrow R^{\dagger}(U) \in \tau$$

where † denotes the relational converse.

Notation 3.2.5. For shorthand, we denote the topology (X, τ) as X^{τ}. As special cases, we denote the discrete
topology on X as $X^{\star}$, and the indiscrete topology X°.

The symmetric monoidal structure is that of product topologies on objects, and products of relations on mor-
phisms.

Reminder 3.2.1 (Topological Space). A *topological space* is a pair
(X, τ), where X is a set, and $\tau \subset \mathcal{P}(X)$ are the *open sets* of X, such that:

"nothing" and "everything" are open

$$\emptyset, X \in \tau$$

Arbitrary unions of opens are open

$$\{U_i : i \in I\} \subseteq \tau \Rightarrow \bigcup_{i \in I} U_i \in \tau$$

Finite intersections of opens are open $n \in \mathbb{N}$:

$$U_1, \cdots, U_n \in \tau \Rightarrow \bigcap_{1 \cdots, i, \cdots n} U_i \in \tau$$

Reminder 3.2.2 (Relational Converse). Recall that a relation $R : S \to T$ is a subset $R \subseteq S \times T$.

$$R^{\dagger} : T \to S := \{(t, s) : (s, t) \in R\}$$

Reminder 3.2.3 (Continuous function). A function between sets
$f : X \to Y$ is a continuous function between topologies $f : (X, \tau) \to$
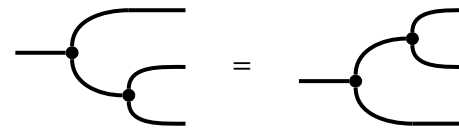
3.3 *TopRel* diagrammatically

3.3.1 *Relations that are always continuous*

HERE ARE FIVE CONTINUOUS RELATIONS FOR ANY X^τ:

everything delete copy

nothing (state) nothing (test)
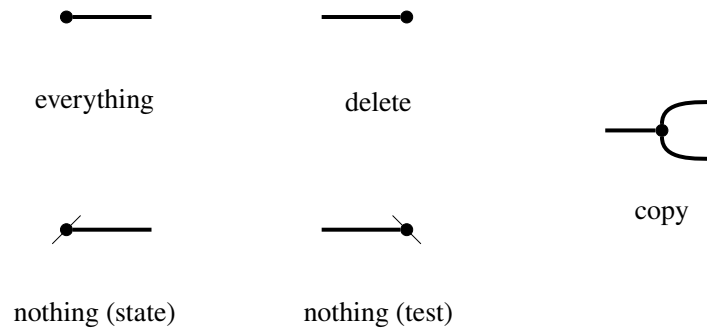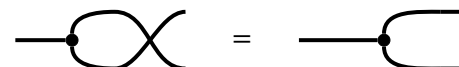
COPY AND DELETE OBEY THE FOLLOWING EQUALITIES:

coassociativity

cocommutativity

counitality

THE COPY MAP CAN ALSO BE USED TO DISTINGUISH THE DETERMINISTIC MAPS – POINTS AND FUNCTIONS – WHICH WE NOTATE WITH AN EXTRA DOT.

point

function
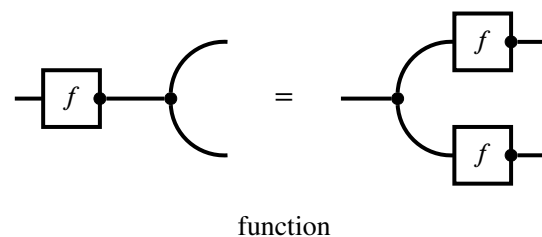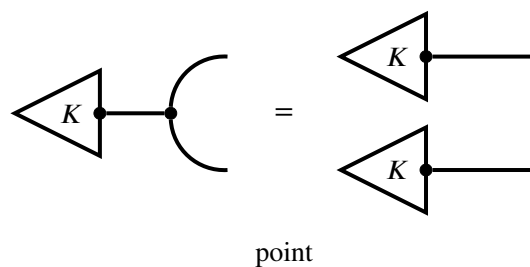
EVERYTHING, DELETE, NOTHING-STATES AND NOTHING-TESTS COMBINE TO GIVE TWO NUMBERS, ONE AND ZERO. There are extra expressions in grey squares above: they anticipate the tape-diagrams we will later use to graphically express another monoidal product of **TopRel**, the direct sum $\oplus$.

(b)one
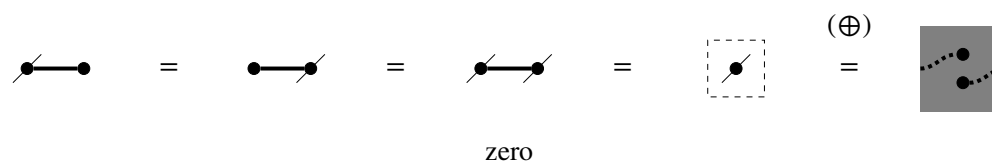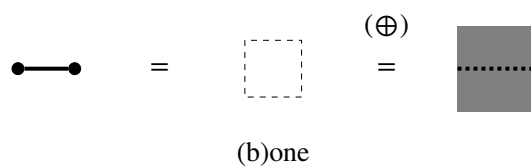
zero

ZERO SCALARS TURN ENTIRE DIAGRAMS INTO ZERO MORPHISMS. There is a zero-morphism for every input-output pair of objects in **TopRel**.

$$\forall X^\tau \forall Y^\sigma \forall f$$

zero

3.4   Continuous Relations by examples

Let's consider three topological spaces and examine the continuous relations between them. This way we can build up intuitions, and prove some tool results in the process.

The **singleton space** consists of a single point which is both open and closed. We denote this space •. Concretely, the underlying set and topology is

$$(\{\star\}\,,\,\{\{\star\},\varnothing\})$$

Open $\longrightarrow$ $\longleftarrow$ Closed
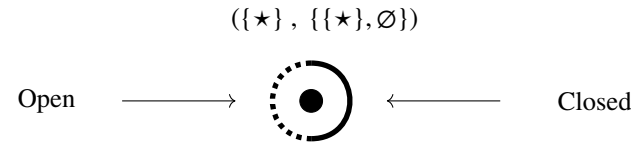
The **Sierpiński space** consists of two points, one of which (in yellow) is open, and the other (in cyan) is closed. We denote this space $\mathcal{S}$. Concretely, the underlying set and topology is:

$$\big(\{0,1\}\,,\,\{\varnothing,\{1\},\{0,1\}\}\big)$$

Open Closed
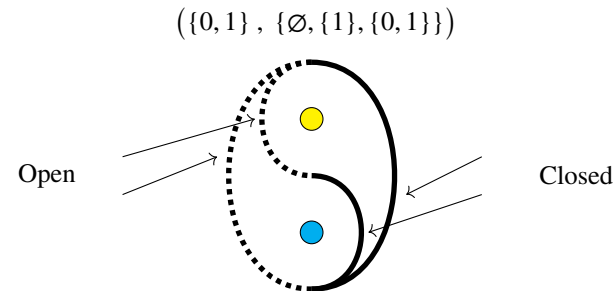
The **unit square** has $[0,1] \times [0,1]$ as its underlying set. Open sets are "blobs" painted with open balls. Points, lines, and bounded shapes are closed. We denote this space ■.

Open Closed
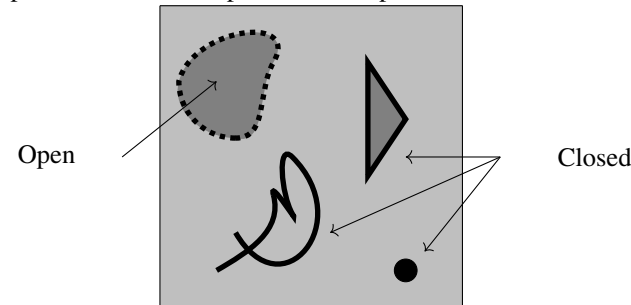
• → •: There are two relations from the singleton to the singleton; the identity relation $\{(\bullet,\bullet)\}$, and the empty relation $\varnothing$. Both are topological.

• → $\mathcal{S}$: There are four relations from the singleton to the Sierpiński space, corresponding to the subsets of $\mathcal{S}$. All of them are topological.

$S \to \bullet$: There four candidate relations from the Sierpiński space to the singleton, but as we see in Example 3.4.1, not all of them are topological.

NOW WE NEED SOME ABSTRACTION. We cannot study the continuous relations between the singleton and the unit square case by case. We discover that continuous relations out of the singleton indicate arbitrary subsets, and that continuous relations into the singleton indicate arbitrary opens.

$\bullet \to \blacksquare$: Proposition 3.4.3 tells us that there are as many continuous relations from the singleton to the unit square as there are subsets of the unit square.

$\blacksquare \to \bullet$: Proposition 3.4.4 tells us that there are as many continuous relations from the unit square to the singleton as there are open sets of the unit square.

THERE ARE 16 CANDIDATE RELATIONS $S \to S$ TO CHECK. A case-by-case approach won't scale, so we could instead identify the building blocks of continuous relations with the same source and target space.

WHICH RELATIONS $X^\tau \to Y^\sigma$ ARE ALWAYS CONTINUOUS?

THE EMPTY RELATION IS ALWAYS CONTINUOUS.

Proposition 3.4.6. *Proof.* The preimage of the empty relation is always $\varnothing$, which is open by definition. □

FULL RELATIONS ARE ALWAYS CONTINUOUS

Proposition 3.4.8. *Proof.* The preimage of any subset of Y – open or not – under the full relation is the whole of X, which is open by definition. □

FULL RELATIONS RESTRICTED TO OPEN SETS IN THE DOMAIN ARE CONTINUOUS.

Proposition 3.4.9. Given an open $U \subseteq X^\tau$, and an arbitrary subset $K \subset Y^\sigma$, the relation $U \times K \subseteq X \times Y$ is open.

Proof. Consider an arbitrary open set $V \in \sigma$. Either V and K are disjoint, or they overlap. If they are disjoint, the preimage of V is $\varnothing$, which is open. If they overlap, the preimage of V is U, which is open. □

CONTINUOUS FUNCTIONS ARE ALWAYS CONTINUOUS.

Proposition 3.4.10. If $f : X^\tau \to Y^\sigma$ is a continuous function, then it is also a continuous relation.

Example 3.4.1 (A noncontinuous relation). The relation $\{(0,\bullet)\} \subset S \times \bullet$ is not a continuous relation: the preimage of the open set $\{\bullet\}$ under this relation is the non-open set $\{0\}$.

Terminology 3.4.2. Call a continuous relation $\bullet \to X^\tau$ a **state** of X^τ, and a continuous relation $X^\tau \to \bullet$ a **test** of X^τ.

Proposition 3.4.3. States $R : \bullet \to X^\tau$ correspond with subsets of X.

Proof. The preimage $R^\dagger(U)$ of a (non-$\varnothing$) open $U \in \tau$ is $\star$ if $R(\star) \cap U$ is nonempty, and $\varnothing$ otherwise. Both $\star$ and $\varnothing$ are open in $\{\star\}^\bullet$. $R(\star)$ is free to specify any non-$\varnothing$ subset of X. The empty relation handles $\varnothing$ as an open of X^τ. □

Proposition 3.4.4. Tests $R : X^\tau \to \bullet$ correspond with open sets $U \in \tau$.

Proof. The preimage $R^\dagger(\star)$ of $\star$ must be an open set of X^τ by definition 3.2.4. $R^\dagger(\star)$ is free to specify any open set of X^τ. □

Reminder 3.4.5 (Empty relation). The **empty relation** $X \to Y$ relates nothing. It is defined:
$$\varnothing \subset X \times Y$$

Reminder 3.4.7 (Full relation). The **full relation** $X \to Y$ relates everything to everything. It is all of $X \times Y$.

Proof. Functions are special cases of relations. The relational converse of a function viewed in this way is the same thing as the preimage. □

THE IDENTITY RELATION IS ALWAYS CONTINUOUS. The identity relation is also the "trivial" continuous map from a space to itself, so this also follows from Proposition 3.4.10.

Proposition 3.4.12. *Proof.* The preimage of any open set under the identity relation is itself, which is open by assumption. □

GIVEN TWO CONTINUOUS RELATIONS $R, S : X^\tau \to Y^\sigma$, HOW CAN WE COMBINE THEM?

Proposition 3.4.14. If $R, S : X^\tau \to Y^\sigma$ are continuous relations, so are $R \cap S$ and $R \cup S$.

Proof. Replace $\square$ with either $\cup$ or $\cap$. For any non-$\varnothing$ open $U \in \sigma$:

$$(R\square S)^\dagger(U) = R^\dagger(U)\square S^\dagger(U)$$

As R, S are continuous relations, $R^\dagger(U), S^\dagger(U) \in \tau$, so $R^\dagger(U)\square S^\dagger(U) = (R\square S)^\dagger(U) \in \tau$. Thus $R\square S$ is also a continuous relation. □

Corollary 3.4.15. Continuous relations $X^\tau \to Y^\sigma$ are closed under arbitrary union and finite intersection. Hence, continuous relations $X^\tau \to Y^\sigma$ form a topological space where each continuous relation is an open set on the base space $X \times Y$, where the full relation $X \to Y$ is "everything", and the empty relation is "nothing".

A TOPOLOGICAL BASIS FOR SPACES OF CONTINUOUS RELATIONS

Definition 3.4.17 (Partial Functions). A **partial function** $X \to Y$ is a relation for which each $x \in X$ has at most a single element in its image. In particular, all functions are special cases of partial functions, as is the empty relation.

Lemma 3.4.18 (Partial functions are a $\cap$-ideal). The intersection $f \cap R$ of a partial function $f : X \to Y$ with any other relation $R : X \to Y$ is again a partial function.

Proof. Consider an arbitrary $x \in X$. $R(x) \cap f(x) \subseteq f(x)$, so the image of x under $f \cap R$ contains at most one element, since $f(x)$ contains at most one element. □

Lemma 3.4.19 (Any single edge can be extended to a continuous partial function). Given any $(x, y) \in X \times Y$, there exists a continuous partial function $X^\tau \to Y^\sigma$ that contains (x, y).

Reminder 3.4.11 (Identity relation). The **identity relation** $X \to X$ relates anything to itself. It is defined:

$$\{(x, x) : x \in X\} \subseteq X \times X$$

Reminder 3.4.13 (Union, intersection, and ordering of relations). Recall that relations $X \to Y$ can be viewed as subsets of $X \times Y$. So it makes sense to speak of the union and intersection of relations, and of partially ordering them by inclusion.

Reminder 3.4.16 (Topological Basis). $\mathfrak{b} \subseteq \tau$ is a basis of the topology τ if every $U \in \tau$ is expressible as a union of elements of $\mathfrak{b}$. Every topology has a basis (itself). Minimal bases are not necessarily unique.
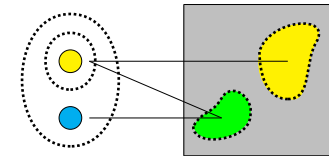
Figure 3.1: Regions of ■ in the image of the yellow point alone will be coloured yellow, and regions in the image of both yellow and cyan will be coloured green:

Proof. Let $\mathcal{N}(x)$ denote some open neighbourhood of x with respect to the topology τ. Then $\{(z, y) : z \in \mathcal{N}(x)\}$ is a continuous partial function that contains (x, y). $\square$

Proposition 3.4.20. Continuous partial functions form a topological basis for the space $(X \times Y)^{(\tau \multimap \sigma)}$, where the opens are continuous relations $X^\tau \to Y^\sigma$.

Proof. We will show that every continuous relation $R : X^\tau \to Y^\sigma$ arises as a union of partial functions. Denote the set of continuous partial functions $\mathfrak{f}$. We claim that:

$$R = \bigcup_{F \in \mathfrak{f}} (R \cap F)$$

The $\supseteq$ direction is evident, while the $\subseteq$ direction follows from Lemma 3.4.19. By Lemma 3.4.18, every $R \cap F$ term is a partial function, and by Corollary 3.4.15, continuous. $\square$

$\mathcal{S} \to \mathcal{S}$: We can use Proposition 3.4.20 to write out the topological basis of continuous partial functions, from which we can take unions to find all the continuous relations, which we depict in Figure 3.5.

$\mathcal{S} \to \blacksquare$: Now we use the colour convention of the points in $\mathcal{S}$ to "paint" continuous relations on the unit square "canvas", as in Figures 3.1 and 3.2. So each continuous relation is a painting, and we can characterise the paintings that correspond to continuous relations $\mathcal{S} \to \blacksquare$ in words as follows: Cyan only in points and lines, and either contained in or at the boundary of yellow or green. Have as much yellow and green as you like.

$\blacksquare \to \mathcal{S}$: The preimage of all of $\mathcal{S}$ must be an open set. So the painting cannot have stray lines or points outside of blobs. The preimage of yellow must be open, so the union of yellow and green in the painting cannot have stray lines or points outside of blobs. Point or line gaps within blobs are ok. Each connected blob can contain any colours in any shapes, subject to the constraint that if cyan appears anywhere, then either yellow or green must occur somewhere. Open blobs with no lines or points outside. Yellow and green considered alone is a painting made of blobs with no stray lines or points. If cyan appears anywhere, then either yellow or green have to appear somewhere.
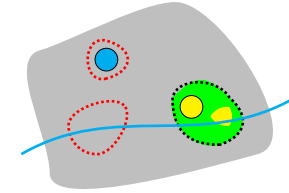
Figure 3.2: Regions in the image of the cyan point alone cannot be open sets by continuity, so they are either points or lines. Points and lines in cyan must be surrounded by an open region in either yellow or green, or else we violate continuity (open sets in red).
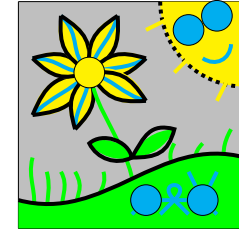
Figure 3.3: A continuous relation $\mathcal{S} \to \blacksquare$: "Flower and critter in a sunny field".
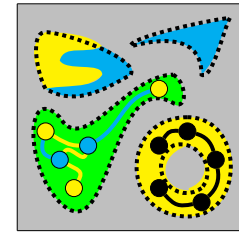
Figure 3.4: A continuous relation $\blacksquare \to \mathcal{S}$: "still math?". Black lines and dots indicate gaps.
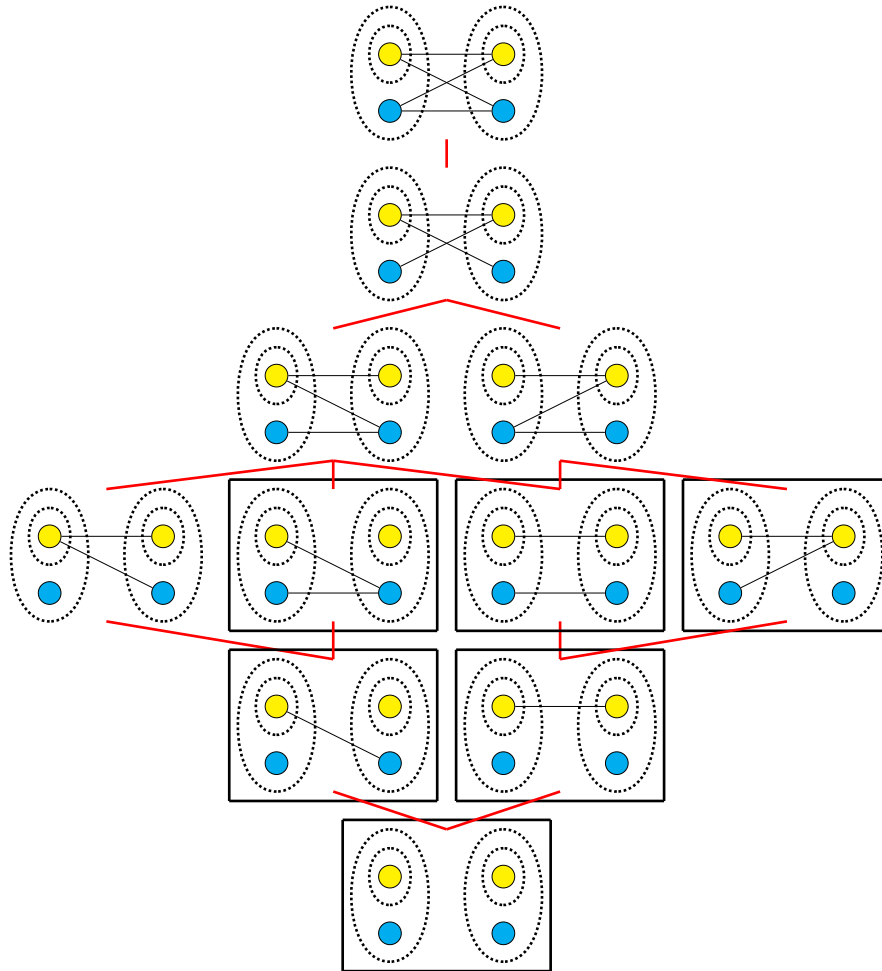
Figure 3.5: Hasse diagram of all continuous relations from the Sierpiński space to itself. Each relation is depicted left to right, and inclusion order is bottom-to-top. Relations that form the topological basis are boxed.

ONE MORE EXAMPLE FOR FUN: $[0, 1] \to \blacksquare$: We know how continuous functions from the unit line into the unit square look.

THEN WHAT ARE THE PARTIAL CONTINUOUS FUNCTIONS? If we understand these, we can obtain all continuous relations by arbitrary unions of the basis. Observe that the restriction of any continuous function to an open set in the source is a continuous partial function. The open sets of $[0, 1]$ are collections of open intervals, each of which is homeomorphic to $(0, 1)$, which is close enough to $[0, 1]$.

ANY PAINTING IS A CONTINUOUS RELATION $[0, 1] \to \blacksquare$. By colour-coding $[0, 1]$ and controlling brushstrokes, we can do quite a lot. Now we would like to develop the abstract machinery required to *formally* paint pictures with words.

Figure 3.6: continuous functions $[0, 1] \to \blacksquare$ follow the naïve notion of continuity: a line one can draw on paper without lifting the pen off the page.

Figure 3.7: So a continuous partial function is "`(countably) many (open-ended) lines, each of which one can draw on paper without lifting the pen off the page.`"
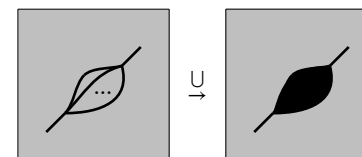
Figure 3.8: We can control the thickness of the brushstroke, by taking the union of (uncountably) many lines.

Figure 3.9: Assign the visible spectrum of light to $[0, 1]$. Colour open sets according to perceptual addition of light, computing brightness by normalising the measure of the open set.

Figure 3.10: Like it or not, a continuous relation $[0, 1] \to \blacksquare$: "The Starry Night", by Vincent van Gogh.

3.5   Populating space with shapes using sticky spiders

3.5.1   When does an object have a spider (or something close to one)?

Example 3.5.2 (The copy-compare spiders of **Rel** are not always continuous)**.** The compare map for the Sierpiński space is not continuous, because the preimage of $\{0, 1\}$ is $\{(0, 0), (1, 1)\}$, which is not open in the product space of S with itself.

Proposition 3.5.3. The copy map is a spider iff the topology is discrete.

Reminder 3.5.1 (copy-compare spiders of **Rel**). For a set X, the *copy* map $X \rightarrow X \times X$ is defined:

$$\{(x, (x, x)) : x \in X\}$$

the *compare* map $X \times X \rightarrow X$ is defined:

$$\{((x, x), x) : x \in X\}$$

These two maps are the (co)multiplications of special frobenius algebras. The (co)units are *delete*:

$$\{(x, \star) : x \in X\}$$

and *everything*:

$$\{(\star, x) : x \in X\}$$

Proof. Discrete topologies inherit the usual copy-compare spiders from **Rel**, so we have to show that when the copy map is a spider, the underlying wire must have a discrete topology. Suppose that some wire has a spider, and construct the following open set using an arbitrary point p:
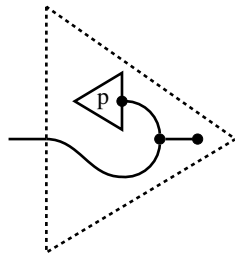
It will suffice to show that this open set is the singleton $\{p\}$ – when all singletons are open, the topology is discrete. As a lemma, using frobenius rules and the property of zero morphisms, we can show that comparing distinct points

Figure 3.11: The generators (in dashed boxes) and relations that make a spider. When the spider satisfies in addition the three inequalities b1-3, we call it a **relation-spider**.

$p \neq q$ yields the $\varnothing$ state.
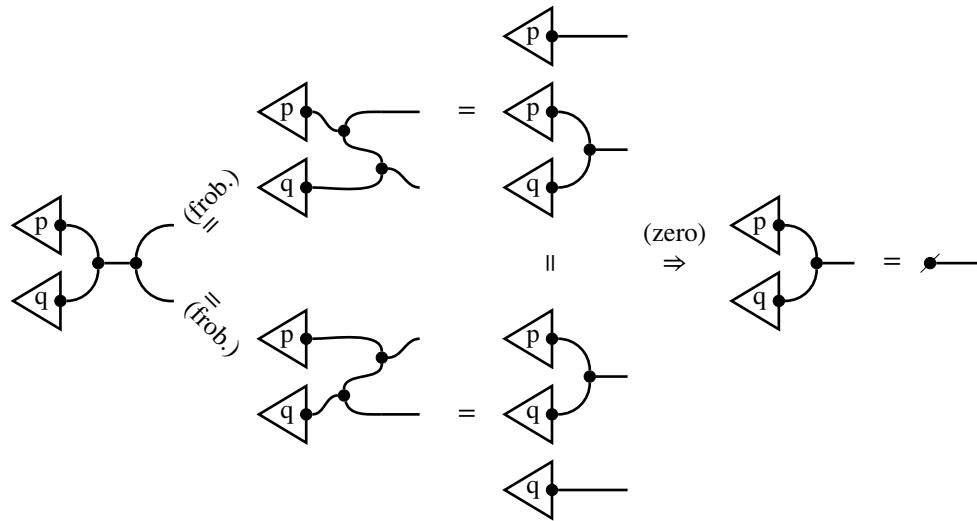
The following case analysis shows that our open set only contains the point p.

$\square$
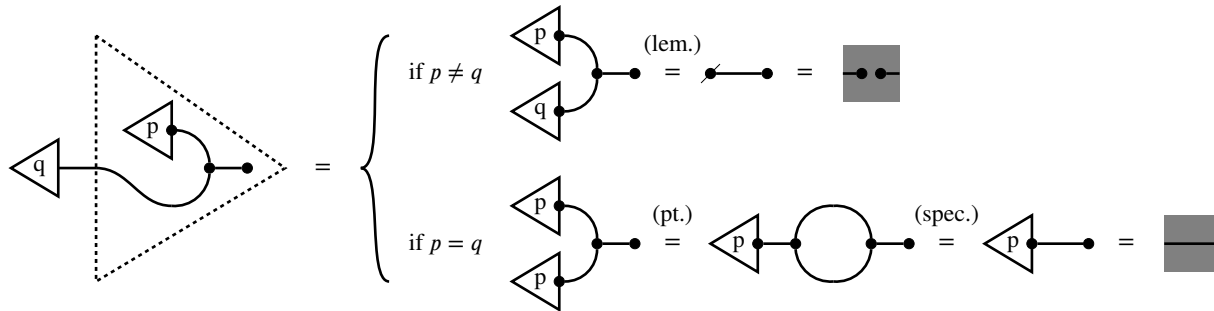
It will be more aesthetic going forward to colour processes and treat the colours as variables instead of labelling them.

WE CAN USE SPLIT IDEMPOTENTS TO TRANSFORM COPY-SPIDERS FROM DISCRETE TOPOLOGIES TO ALMOST-SPIDERS ON OTHER SPACES. We can graphically express the behaviour of a split idempotent e as follows, where the semicircles for the section and retract r, s form a visual pun.

Reminder 3.5.4 (Split idempotents). An **idempotent** in a category is a map $e : A \to A$ such that

$$A \xrightarrow{e} A \xrightarrow{e} A = A \xrightarrow{e} A$$

A **split idempotent** is an idempotent $e : A \to A$ along with a **retract** $r : A \to B$ and a **section** $s : B \to A$ such that:

$$A \xrightarrow{e} A = A \xrightarrow{r} B \xrightarrow{s} A$$

$$B \xrightarrow{s} A \xrightarrow{r} B = B \xrightarrow{id} B$$

Definition 3.5.5 (Sticky spiders). A **sticky spider** (or just an *e*-spider, if we know that *e* is a split idempotent), is a spider *except* every identity wire on any side of an equation in Figure 3.11 is replaced by the idempotent *e*.

e-(co)unitality

e-special

The desired graphical behaviour of a sticky spider is that one can still coalesce all connected spider-bodies together, but the 1-1 spider "sticks around" rather than disappearing as the identity. This is achieved by the following rules that cohere the idempotent *e* with the (co)unit and (co)multiplications; they are the same as the usual rules for a special commutative frobenius algebra with two exceptions. First, where an identity wire appears in an equation, we replace it with an idempotent. Second, the monoid and comonoid components freely emit and absorb idempotents. By these rules, the usual proof [] for the normal form of spiders follows, except the idempotent becomes an explicit 1-1 spider, rather than the

identity.

Coassociative

Associative

Commutative

Cocommutative

Frobenius

Sticky-special

Sticky-(co)unital

Absorb-comult

Absorb-mult

Absorb-(co)unit
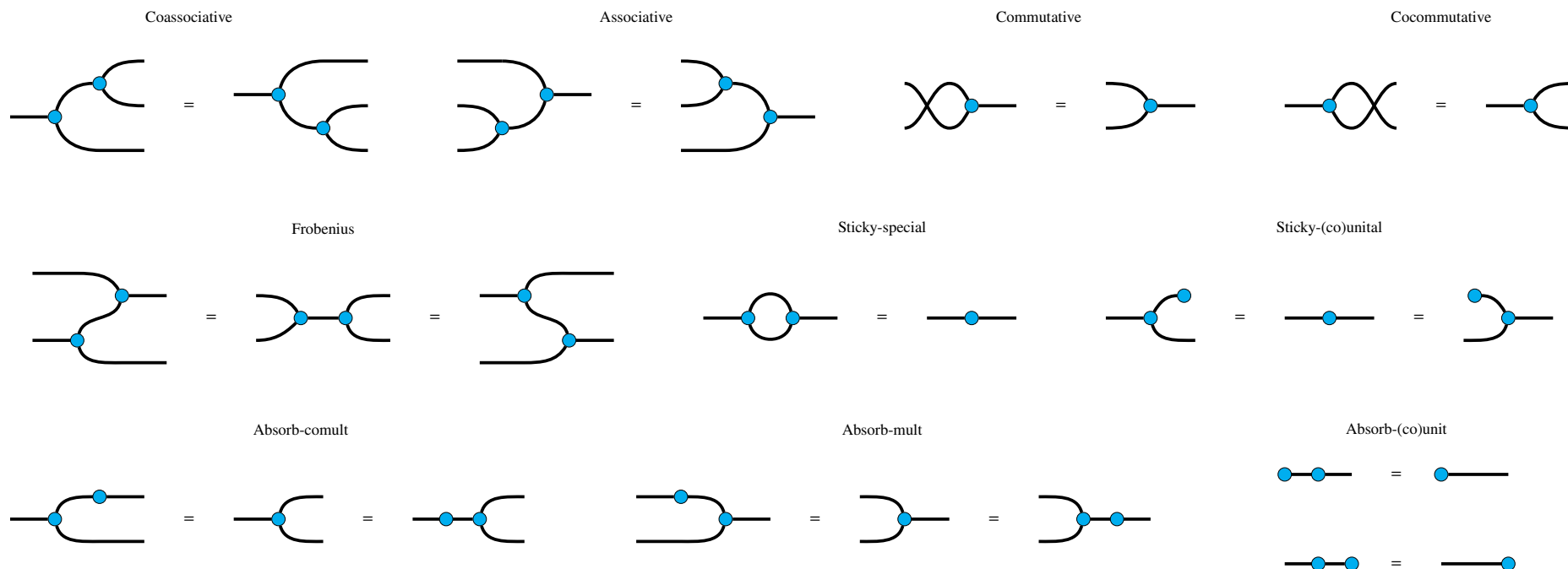
Construction 3.5.6 (Sticky spiders from split idempotents)**.** Given an idempotent $e : Y^\sigma \to Y^\sigma$ that splits through a discrete topology $X^\star$, we construct a new (co)multiplication as follows:

Proposition 3.5.7 (Every idempotent that splits through a discrete topology gives a sticky spider)**.**
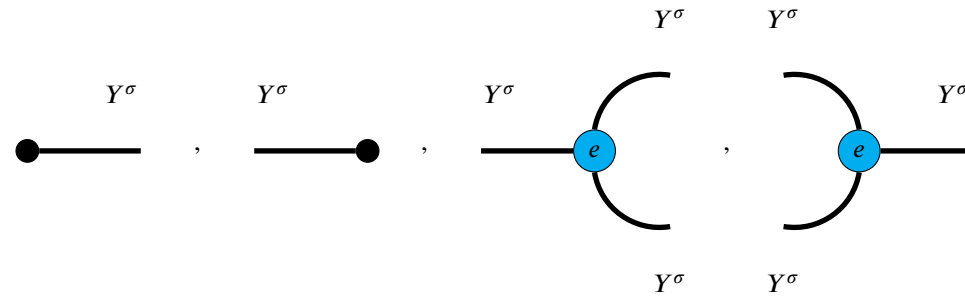
is a sticky spider

Proof. We can check that our construction satisfies the frobenius rules as follows. We only present one equality; the rest follow the same idea.

(defn.)   (idem.)   (frob.)

(split)   (defn.)
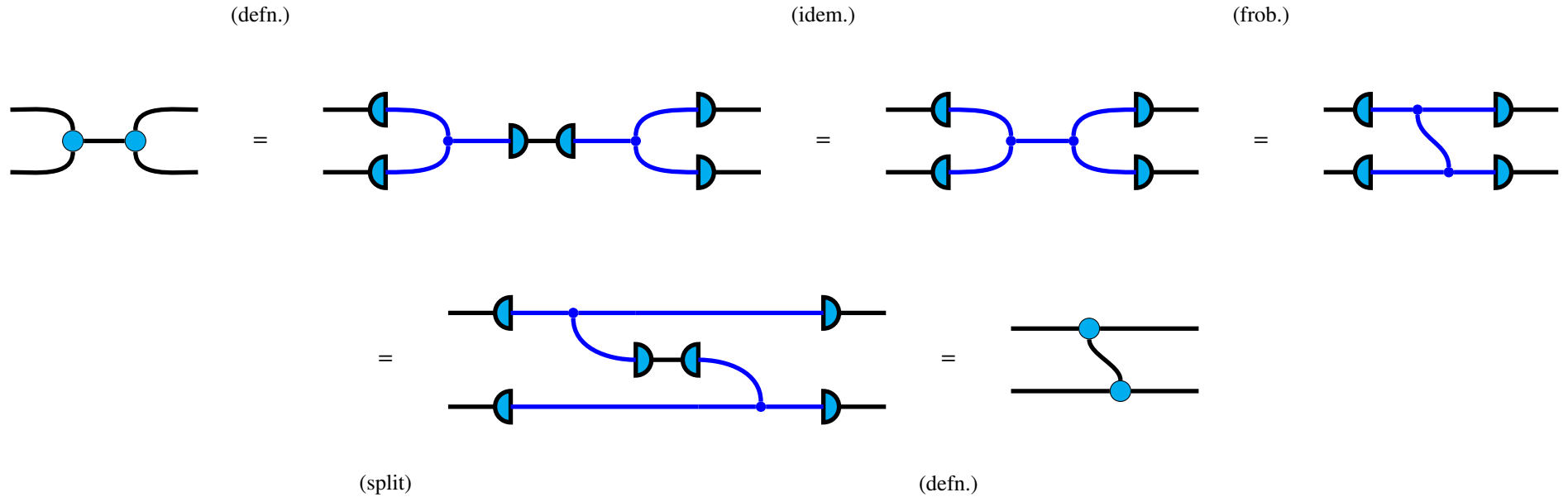
To verify the sticky spider rules, we first observe that since

$$X^\star \xrightarrow{s} Y^\sigma \xrightarrow{r} X^\star = X^\star \xrightarrow{id} X^\star$$

r must have all of $X^\star$ in its image, and s must have all of $X^\star$ in its preimage, so we have the following:

Now we show that e-unitality holds:

The proofs of e-counitality, and e-speciality follow similarly.   □

WE CAN PROVE A PARTIAL CONVERSE OF PROPOSITION 3.5.7: we can identify two diagrammatic equations that tell us precisely when a sticky spider has an idempotent that splits though some discrete topology.

Theorem 3.5.8. A sticky spider has an idempotent that splits through a discrete topology if and only if in addition to the sticky spider equalities, the following relations are also

satisfied.

Unit/everything
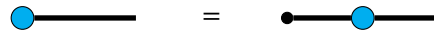
Comult/copy
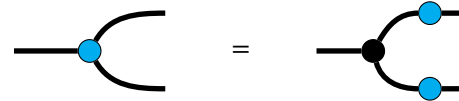
The proof is rather involved, so we provide a map below of the various lemmas and propositions that will yield the claim.

basis decomposition : comultiplication

comult/copy

counit/delete

Prop. 3.5.9

basis decomposition : multiplication

Lemma 3.5.10

Lemma 3.5.18

basis decomposition : idem.

Lemma 3.5.17

Prop. 3.5.11

Prop. 3.5.21

Prop. 3.5.12

Prop. 3.5.13

Prop. 3.5.14

Prop. 3.5.20

Lemma 3.5.15

unit/everything

copiable-basis

basis decomposition : counit
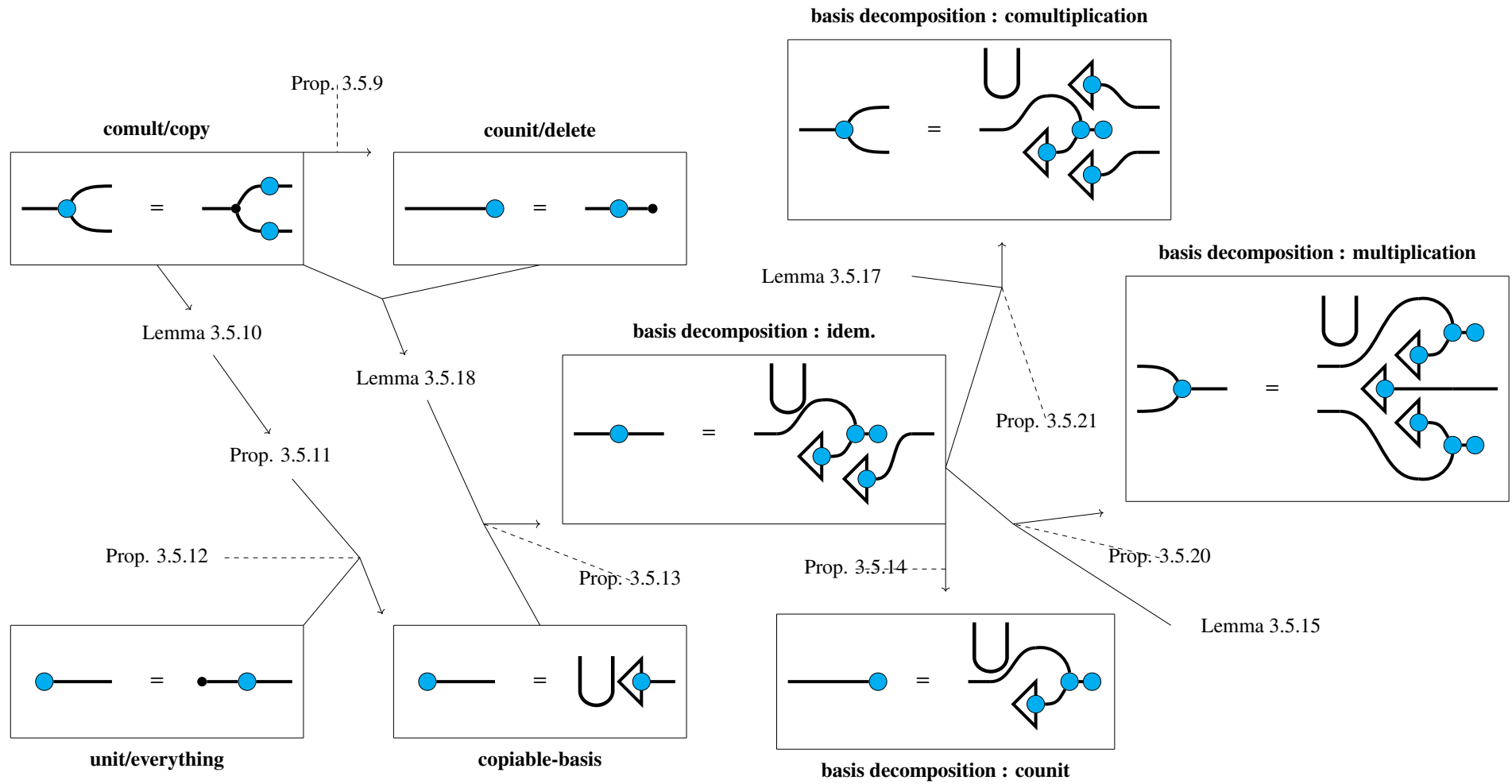
Proposition 3.5.9 (comult/copy implies counit/delete)**.**

Proof.

(comult/copy)   (del)

(del)   (e-unit)   (copy-del)

So:

So:

□

Lemma 3.5.10 (All-or-Nothing). Consider the set $e(\{x\})$ obtained by applying the idempotent e to a singleton $\{x\}$, and take an arbitrary element $y \in e(x)$ of this set. Then $e(\{y\}) = \varnothing$ or $e(\{x\}) = e(\{y\})$. Diagrammatically:

Proof.

Suppose

For the claim, we seek:

We have the following inclusion:

Therefore:

So we have the following equality:

(frob.)   (comult/copy)   (e-spider)   (e-copy)
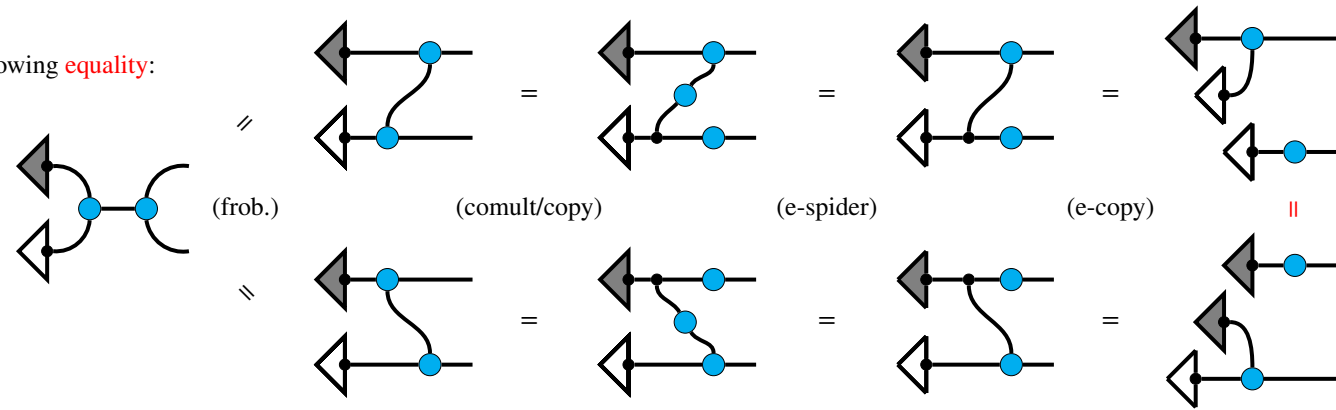
Which implies:
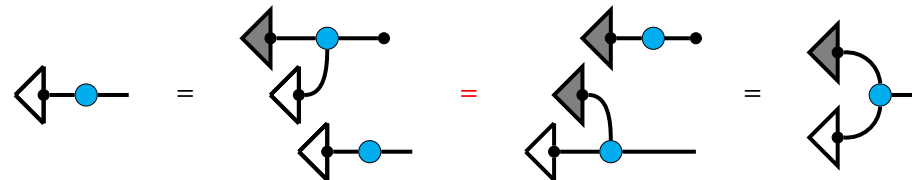
and symmetrically,
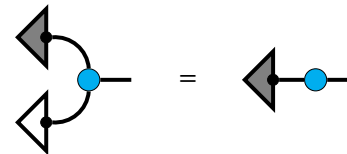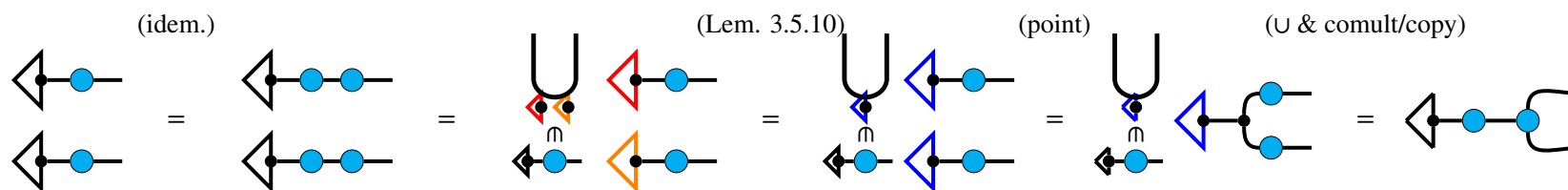
So we have the claim:
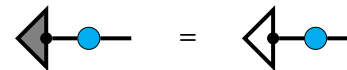
Proposition 3.5.11 (*e* of any point is *e*-copiable)**.**
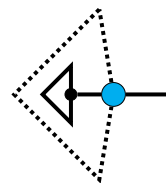
Proof.

□

Proposition 3.5.12 (The unit is the union of all *e*-copiables)**.**

Proof.

The union of *all e*-copiables is a subset of the unit.

(Lem. 3.5.18)   (evr.)   (unit/evr.)

The unit is *some* union of *e*-copiables.

(unit/evr.)   (Prop. 3.5.11)
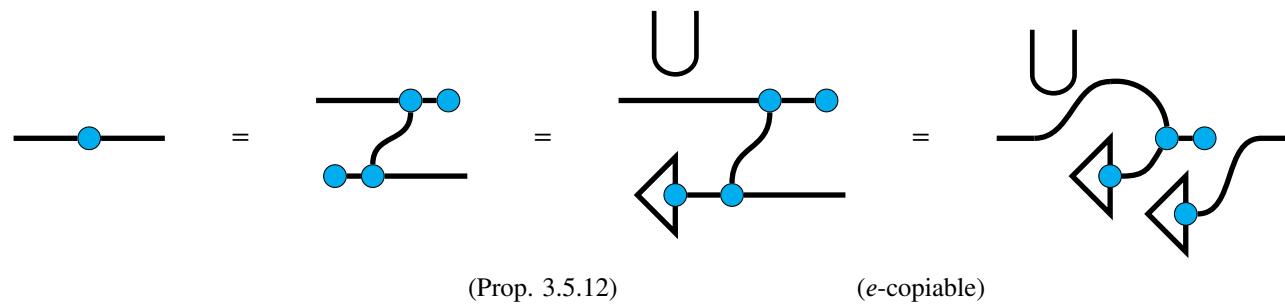
So the containment must be an equality.

□

Proposition 3.5.13 (*e*-copiable decomposition of *e*)**.**

Proof.

(Prop. 3.5.12)   (*e*-copiable)

□
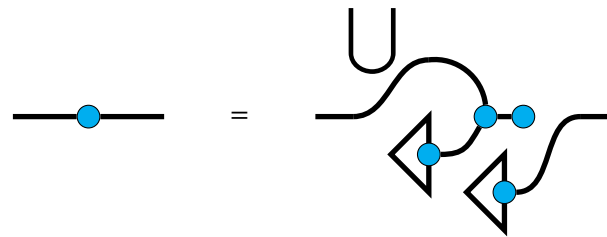
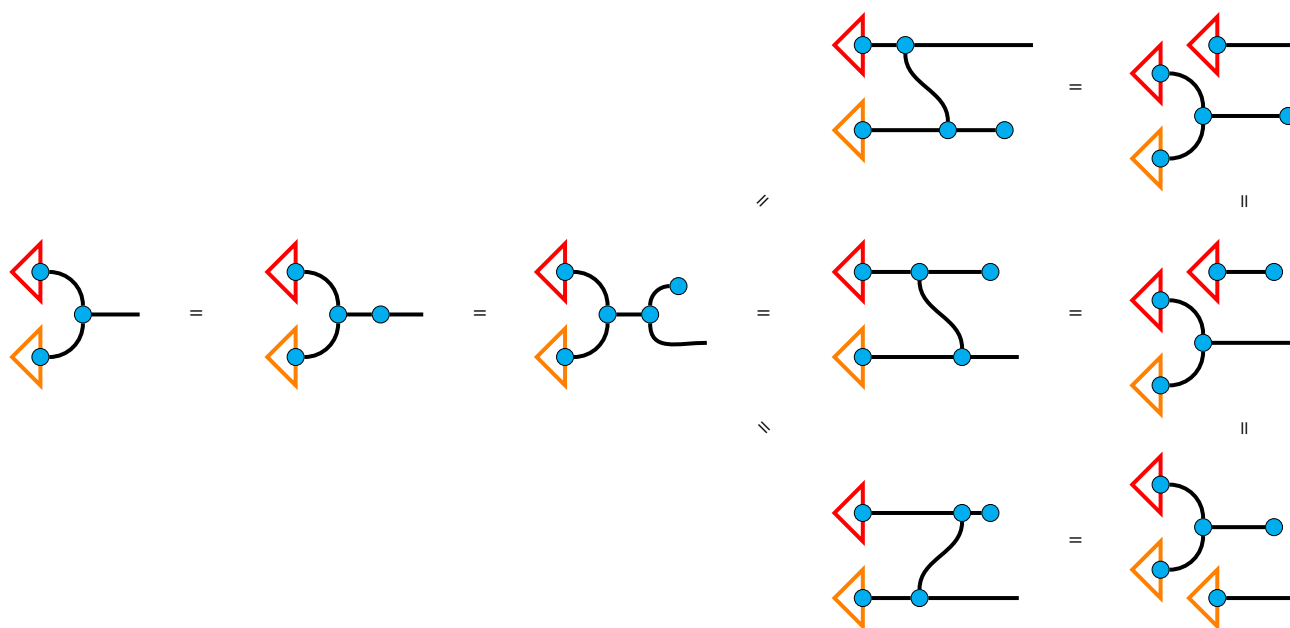**Proposition 3.5.14** (*e*-copiable decomposition of counit)**.**

Proof.

□

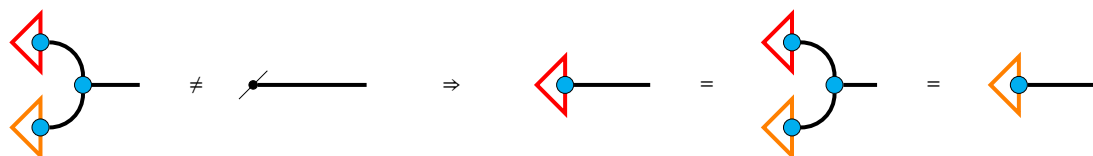THE *e*-COPIABLE STATES REALLY DO BEHAVE LIKE AN ORTHONORMAL BASIS, AS THE FOLLOWING LEMMAS SHOW.

Lemma 3.5.15 (*e*-copiables are orthogonal under multiplication)**.**

Proof.

So:

□

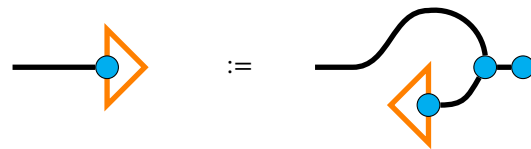**Convention 3.5.16** (Shorthand for the open set associated with an *e*-copiable)**.** We introduce the following diagrammatic shorthand.

Including the coloured dot is justified, because these open sets are co-copiable with respect to the multiplication of the sticky spider.
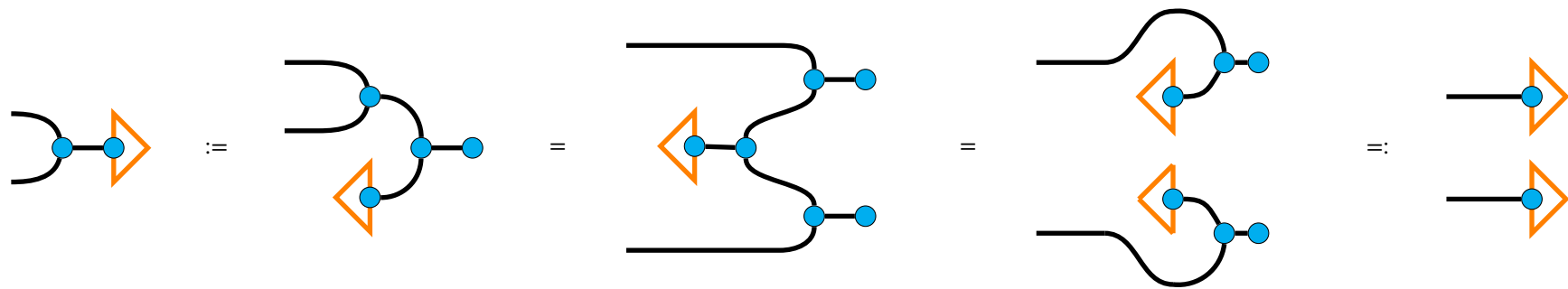
Lemma 3.5.17 (Co-match)**.**

Proof.

The claim then follows by applying Lemma 3.5.15 to the final diagram.   □

Lemma 3.5.18 (e-copiables are e-fixpoints).

Proof.

Observe that the final equation of the proof also holds when the initial e-copiable is the empty set.   □

Lemma 3.5.19 (*e*-copiables are normal)**.**

Proof.

□

Proposition 3.5.20 (*e*-copiable decomposition of multiplication)**.**

Proof.

(e-spider)   (Prop. 3.5.13)   (Lem. 3.5.15)

□
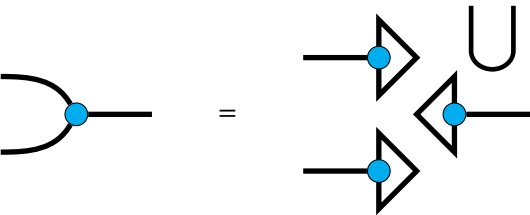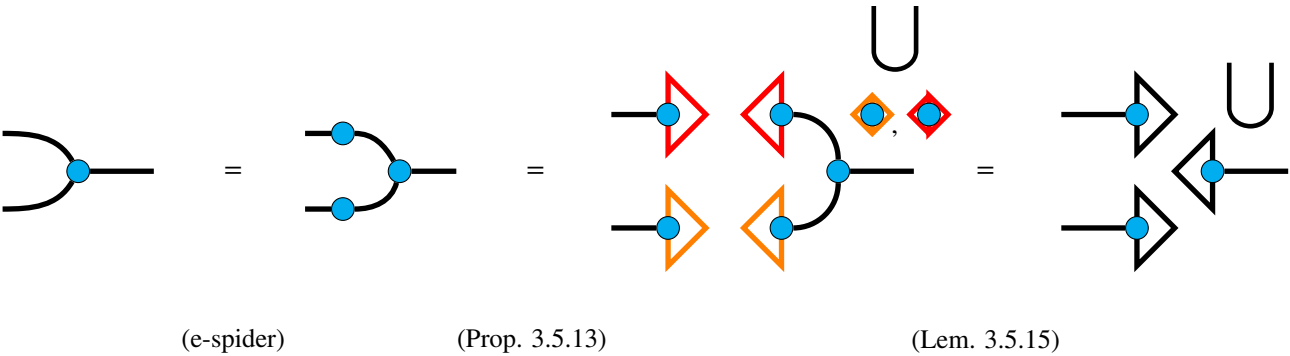
Proposition 3.5.21 (*e*-copiable decomposition of comultiplication)**.**
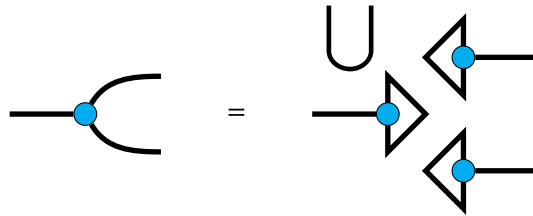
Proof.
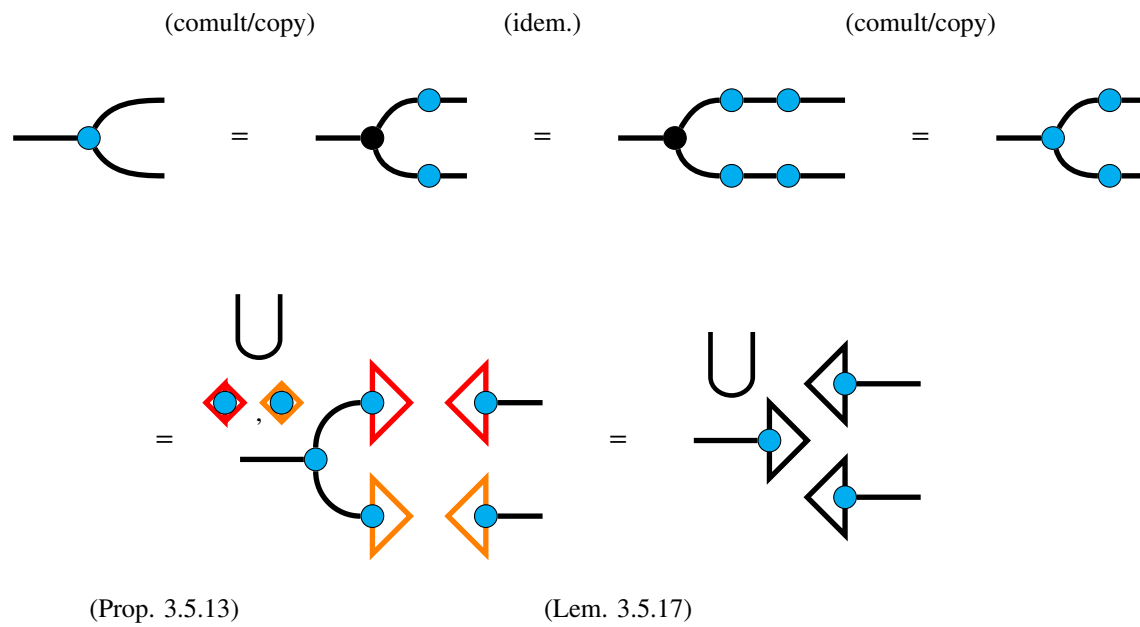
Now we can prove Theorem 3.5.8.

Proof. First a reminder of the claim; we want to show that when given a sticky spider, the following relations hold if and only if the idempotent splits through a discrete topology.
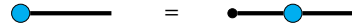
Unit/everything   Comult/copy
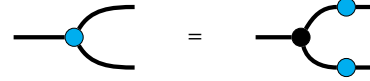
The crucial observation is that the *e*-copiable decomposition of the idempotent given by Proposition 3.5.13 is equivalent to a split idempotent though the set of *e*-copiables equipped with discrete topology.

$$:= \{(x, i) \mid i \in \mathcal{I} , \ x \in |i> \}$$

$$\bigcup_{i \in \mathcal{I}}$$ $$\mathcal{I}^{\star}$$
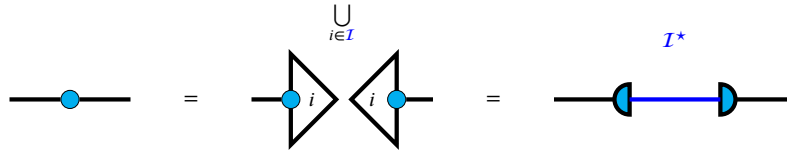
(Prop. X)

$$:= \{(i, x) \mid i \in \mathcal{I} , \ x \in < i| \}$$

By copiable basis Proposition 3.5.12 and the decompositions Propositions 3.5.14, 3.5.20, 3.5.21, we obtain the only-if direction.

(unit/evr.)   (Prop. 3.5.12)   (Prop. 3.5.9)   (Prop. 3.5.14)

(Prop. 3.5.21)   (Prop. 3.5.20)
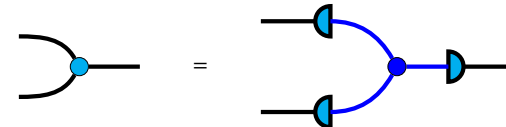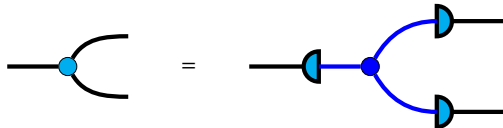
The if direction is an easy check. For the unit/everything relation, we have:

$$
\text{(split)} \qquad\qquad \text{(Prop. 3.5.7)} \qquad\qquad \text{(split)}
$$

For the counit/delete relation, we observe that for any split idempotent, the retract must be a partial function. To see this, suppose the split idempotent $e = r; s$ is on (X, τ) and the discrete topology is $Y^\star$. Seeking contradiction, if the retract is not a partial function, then there is some point $x \in X$ such that $x \in e(x)$, and the image $I := r(x) \subseteq Y$ contains more than one point, which we denote and discriminate $a, b \in r(x) \subseteq Y$ and $a \neq b$. Because the composite $s; r = 1_Y$ of the section and retract must recover the identity on $Y^\star$, the section s must be total – i.e. the image $s(X) = Y$. So $x \in s(a) \cap s(b)$. Now we have that $(a, x), (b, x) \in s$, and $(x, a), (x, b) \in r$, therefore $(a, b), (b, a) \in s; r$, which by $a \neq b$ contradicts that $s; r$ is the identity relation 1_Y.

$$
\text{(split)} \qquad\qquad \text{(pfn.)} \qquad\qquad \text{(split)}
$$

□

3.6   Topological concepts in flatland via **TopRel**

The goal of this section is to demonstrate the use of sticky spiders as formal semantics for the kinds of schematic doodles or cartoons we would like to draw. Throughout we consider sticky spiders on $\mathbb{R}^2$. In Section 3.6.1, we introduce how sticky spiders may be viewed as labelled collections of shapes. In service of defining *configuration spaces* of shapes up to rigid displacement, we diagrammatically characterise the topological subgroup of isometries of $\mathbb{R}^2$ by building up in Sections 3.6.2 and 3.6.3 the diagrammatic presentations of the unit interval, metrics, and topological groups. To further isolate rigid displacements that arise from continuous sliding motion of shapes in the plane (thus excluding displacements that result in mirror-images), in Sections 3.6.4 and 3.6.5 we diagrammatically characterise an analogue of homotopy in the relational setting. Finally, in Sections 3.6.6 and 3.6.7 we build up a stock of topological concepts and study by examples how implementing these concepts within text circuits explains some idiosyncrasies of the theory: namely why noun wires are labelled by their noun, why adjective gates ought to commute, and why verb gates do not.

3.6.1   Shapes and places

Remark 3.6.5. When we draw on a finite canvas representing all of euclidean space, properly there should be a fishbowl effect that relatively magnifies shapes close to the origin and shrinks those at the periphery, but that is only an artefact of representing all of euclidean space on a finite canvas. Since all the usual metrics are still really there, going forward we will ignore this fishbowl effect and just doodle shapes as we see fit.
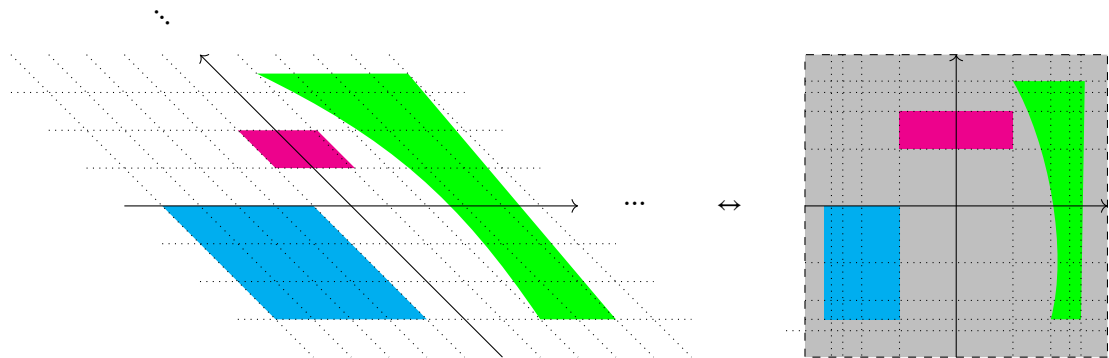
Definition 3.6.1 (Labels, shapes, cores, halos). Recall by Proposition 3.5.13 that we can express the idempotent as a union of continuous relations formed of a state and test, for some indexing set of *labels* $\mathcal{L}$.

A *shape* is a component of this union corresponding to some arbitary $l \in L$. So we refer to a sticky spider as a labelled collection of shapes. The state of a shape is the *halo* of the shape. The halos are precisely the copiables of the sticky spider. The test of a shape is the *core*. The cores are precisely the cocopiables of the sticky spider.
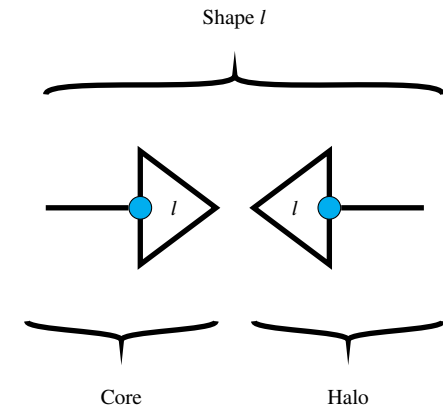
Proposition 3.6.2 (Core exclusion: Distinct cores cannot overlap). *Proof.* A direct consequence of Lemma 3.5.17. □

Proposition 3.6.3 (Core-halo exclusion: Each core only overlaps with its corresponding h Seeking contradiction, if a core overlapped with multiple halos, Lemma 3.5.18 would be violated. □

Proposition 3.6.4 (Halo non-exclusion: halos may overlap). *Proof.* By

Example 3.6.6 (Where is a piece on a chessboard?). How is it that we quotient away the continuous structure of positions on a chessboard to locate pieces among a discrete set of squares? Evidently shifting a piece a little off the centre of a square doesn't change the state of the game, and this resistance to small perturbations suggests that a topological model is appropriate. We construct two spiders, one for pieces, and one for places on the chessboard. We consider the pieces spider to consist of open balls of some radius r, and we consider the places spider to consist of square halos (which tile the chessboard), containing a core inset by the same radius r; in this way, any piece can only overlap at most one square. As a technical aside, to keep the core of the tiles open, we can choose an arbitrarily sharp curvature ϵ at the corners.
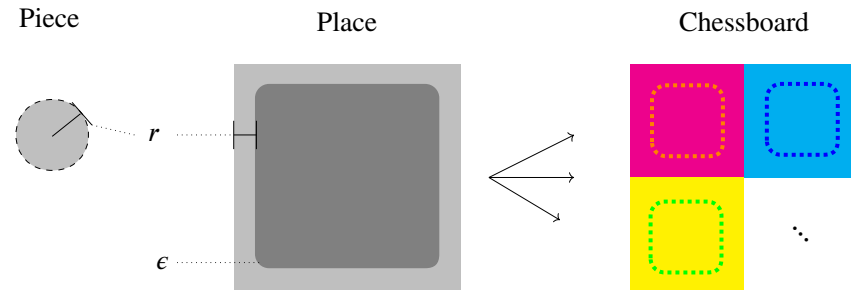
Now we observe that the calculation of positions corresponds to composing sticky spiders. As input, we provide the spatial extents of the white pieces on the board. We can first abstract away the spatial extents of the individual chesspieces by composing with the pieces spider. We can then obtain the set of positions of each piece by composing with the places spider. The composite pieces;places will send the king to a2, the bishop to b4, and the knight to d1.
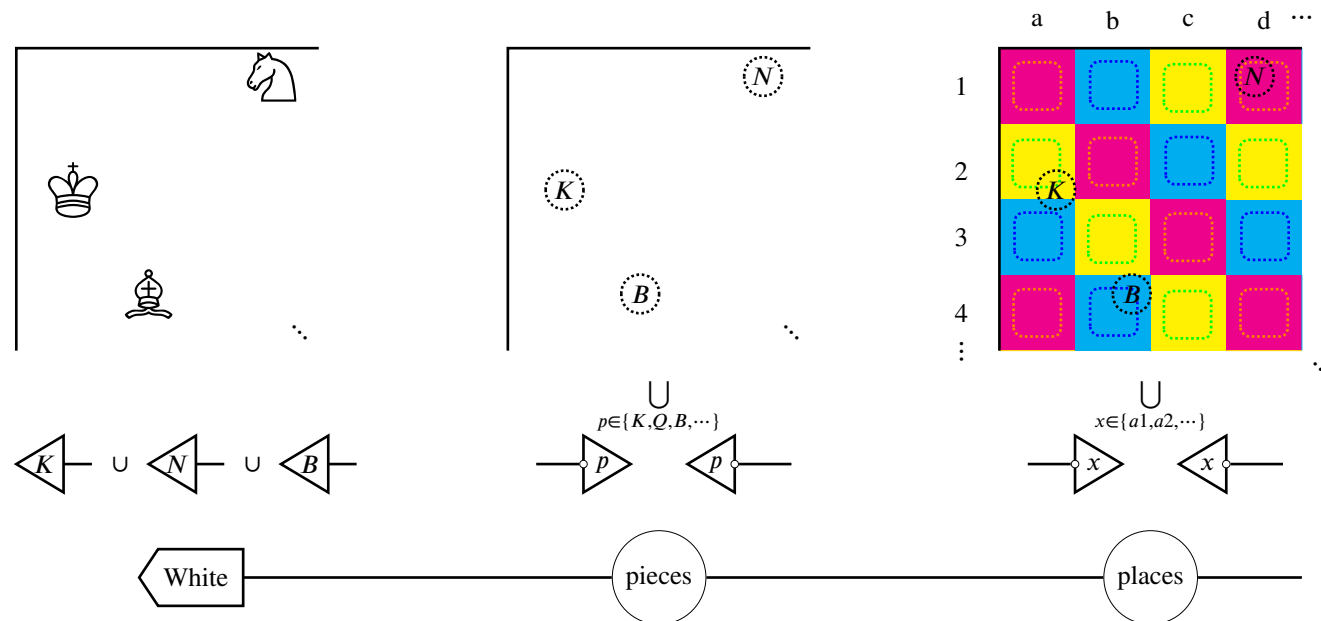
3.6.2   The unit interval

To begin modelling more complex concepts, we first need to extend our topological tools. If we have the unit interval, we can begin to define what it would mean for spaces to be connected (by drawing lines between points in those spaces), and we can also move towards defining motion as movement along a line.

THE REALS There are many spaces homeomorphic to the real line. How do we know when we have one of them? The following theorem provides an answer:

Theorem 3.6.7 (Friedman). Let $\big((X, \tau), <\big)$ be a topological space with a total order. If there exists a continuous map $f : X \times X \to X$ such that $\forall a, b_{\in X} : a < f(a, b) < b$, then X is homeomorphic to $\mathbb{R}$.

   We can define all of these pieces using diagrammatic equations.

LESS THAN We define a total order $<$ as an open set – i.e. a test – on $X \times X$ that obeys the usual axiomatic rules:

Antireflexive $(\forall x : x \not< x)$ Transitive $(\forall xyz : x < y \,\&\, y < z \Rightarrow x < z)$ Antireflexive $\forall xy \neg (x < y \,\&\, y < x)$

Trichotomy $(\forall xy : x < y \ \vee\ y < x \ \vee\ \ x = y)$
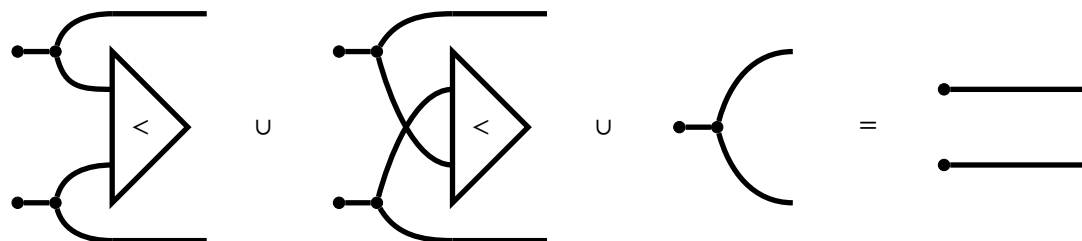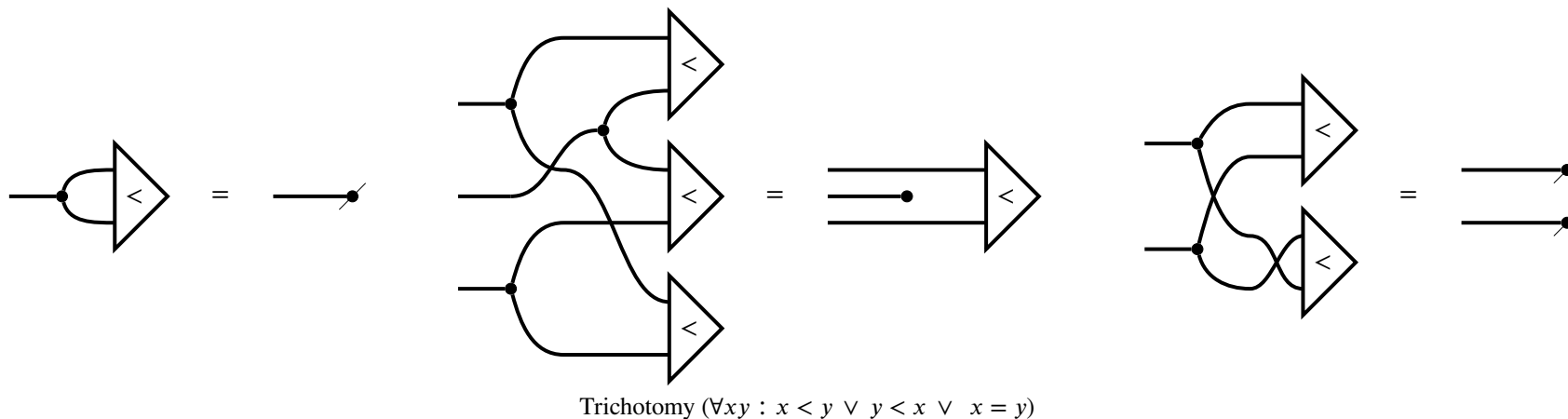
ENDPOINTS We can introduce endpoints for open intervals directly by asking for the space X to have points that are less than or greater than all other points. Another method, which we will use here for primarily aesthetic reasons, is to use endocombinators to define suprema. Endocombinators are like functional expressions applied to diagrams. For a motivating example, consider the case when we have a locally indiscrete topology:

Definition 3.6.8 (Locally indiscrete topology). (X, τ) is *locally indiscrete* when every open set is also closed.

If we know that a topology is locally indiscrete and we are given an open U, we would like to notate the complement X/U – which we know to be open – as any of the following, which only differ up to notation.
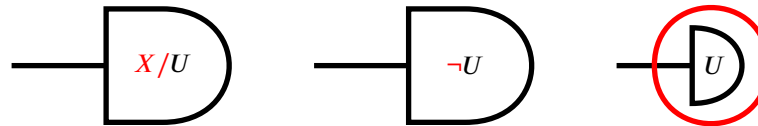
Unfortunately, the complementation operation $X/-$ is not in general a continuous relation, hence in the lattermost expression above we resort to using bubbles as a syntactic sugar. Formally, these bubbles are *endocombinators*, the semantics and notation for which we borrow and modify from [].

Definition 3.6.9 (Partial endocombinator). In a category C, a *partial endocombinator* on a homset $(C)(A, B)$ is a function $(C)(A, B) \to (C)(A, B)$

Using this technology, we can define:
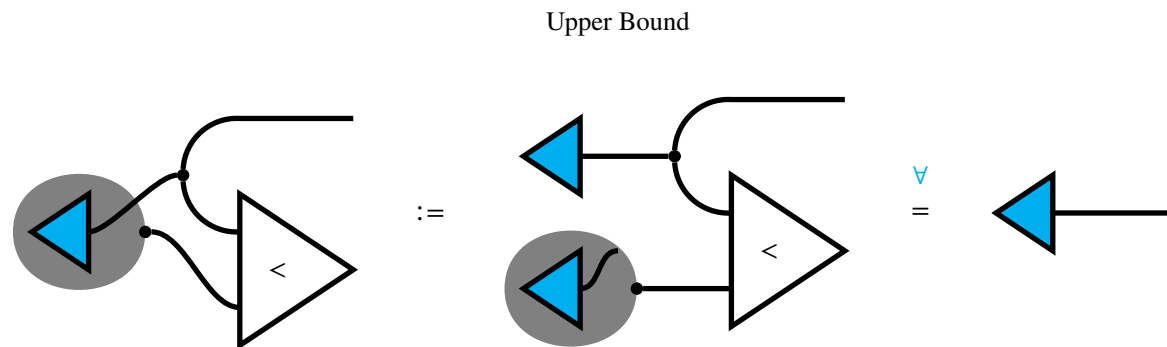
Upper Bound

And we can add in further equations governing the upper bound endocombinator to turn it into a supremum, where the lower endpoint is obtained as the supremum of the empty set, and the upper endpoint is the supremum of the whole set.

Supremum
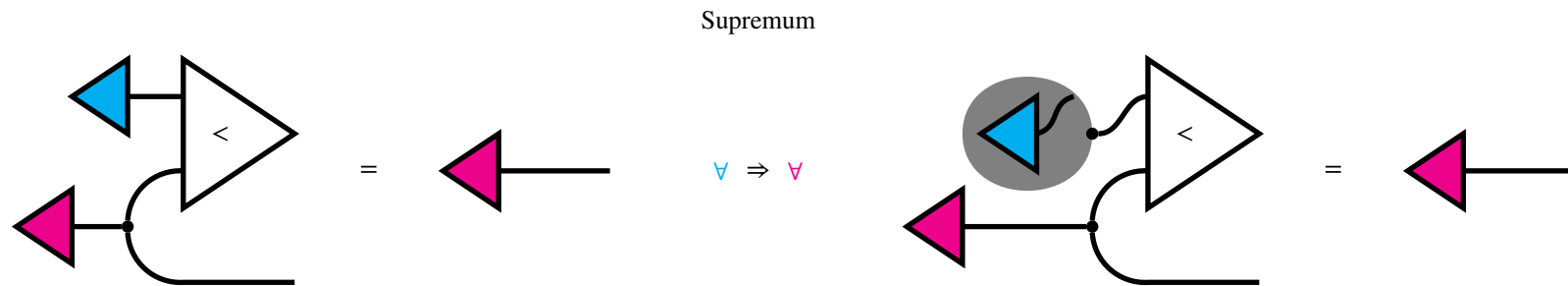
Now we can define endpoints purely graphically:

Lower endpoint

Upper endpoint

Going forward, we will denote the unit interval using a thick dotted wire.

SIMPLY CONNECTED SPACES

Once we have a unit interval, we can define the usual topological notion of a simply connected space: one where any two points can be connected by a continuous line without leaving the space.

V is *simply connected* when:
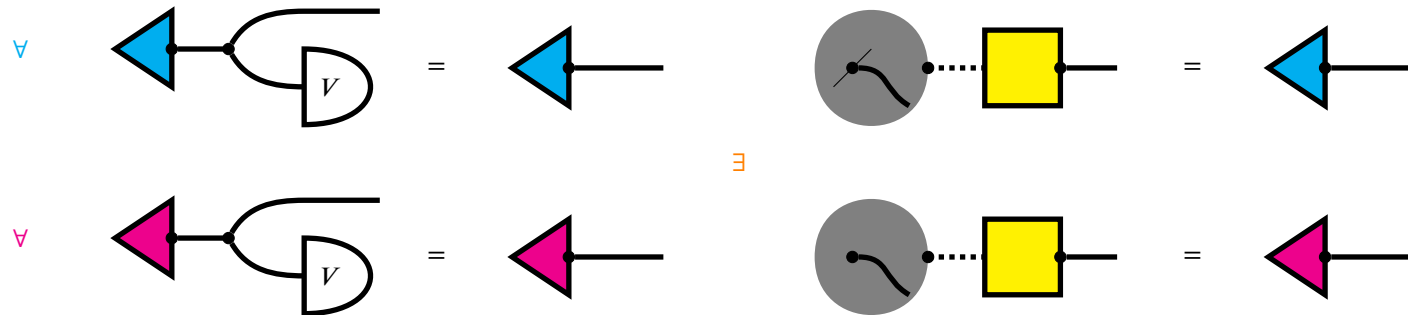
This is a useful enough concept that we will notate simply connected open sets as follows, where the hole is a reminder that simply connected spaces might still have holes in them.
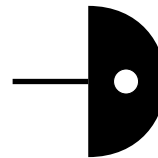
3.6.3   *Displacing shapes*

Static shapes in space are nice, but moving them around would be nicer. So we have to define a stock of concepts to express rigid motion. Rigidity however is a difficult concept to express in topological spaces up to homeomorphism – everyone is well aware of the popular gloss of topology in terms of coffee cups being homeomorphic to donuts. To obtain rigid transformations as we have in Euclidean space, we need to define metrics, and in order to do that, we need addition.

ADDITION

More precisely, we only need an additive monoid structure on the unit interval. We do not care about obtaining precise values from our metric, and we will not need to subtract distances from each other. All we need to know is that the lower endpoint stands in for "zero distance" – as the unit of the monoid – and that adding positive distances together will give you a larger positive distance deterministically.

Associative

Symmetric

Monotone

Unital (with lower endpoint)
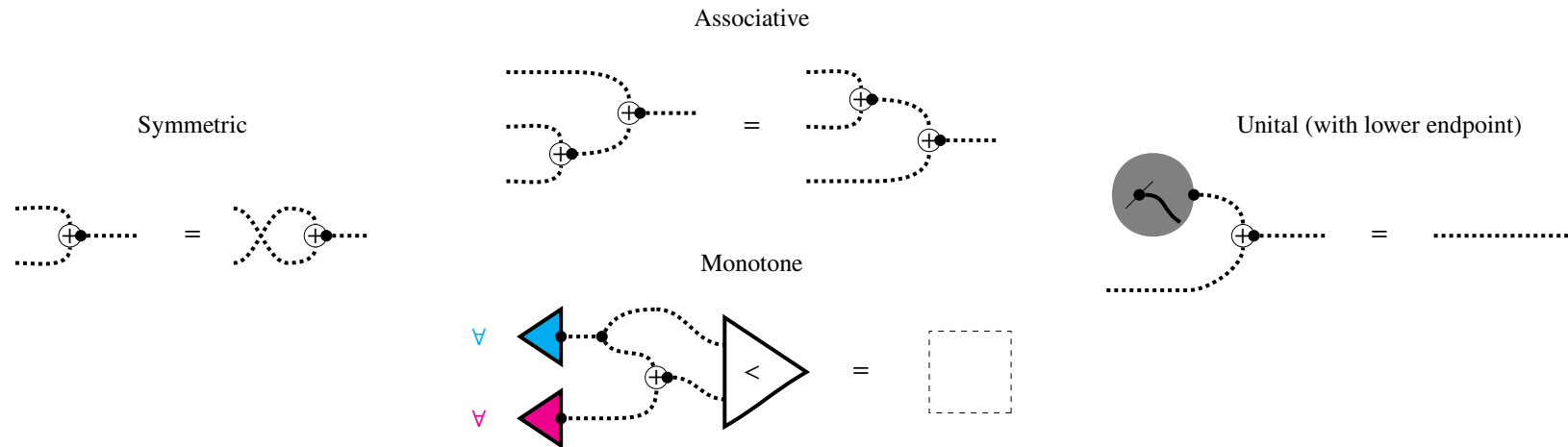
METRICS

A metric on a space is a continuous map $X \to \mathbf{R}^+$ to the positive reals that satisfies the following axioms.

$$d(x, y) = d(y, x)$$

$$d(x, z) \leq d(x, y) + d(y, z)$$

$$d(x, y) = 0 \iff x = y$$
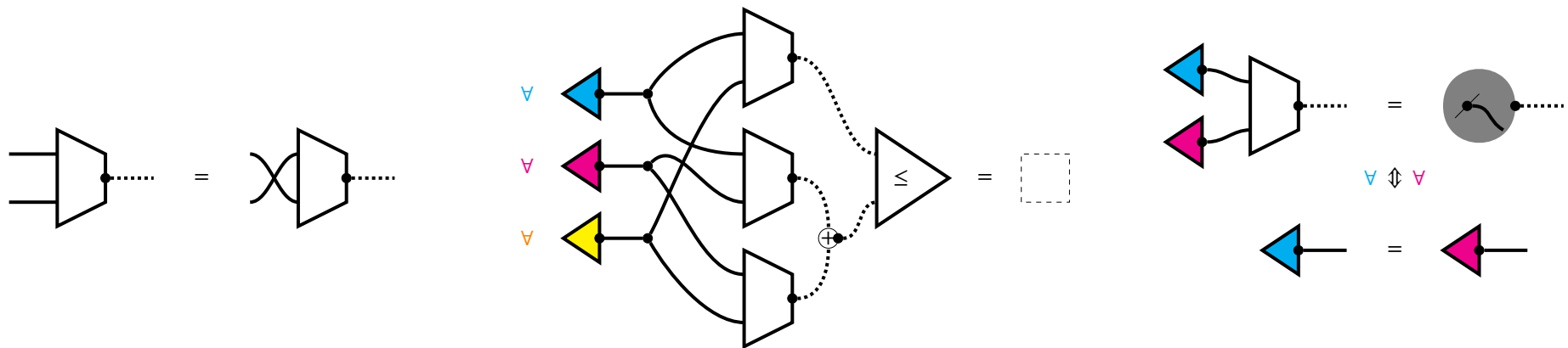
OPEN BALLS

Once we have metrics, we can define the usual topological notion of open balls. Open balls will come in handy later, and a side-effect which we note but do not explore is that open balls form a basis for any metric space, so in the future whenever we construct spaces that come with natural metrics, we can speak of their topology without any further work.

Open ball of radius ϵ at a point **x**
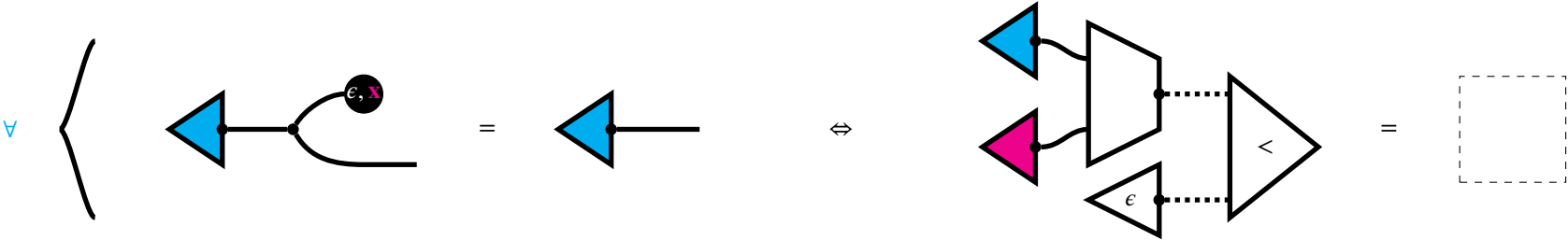
TOPOLOGICAL GROUP

It is no trouble to depict collections of invertible transformations of spaces $X \to X$:

A topological group G
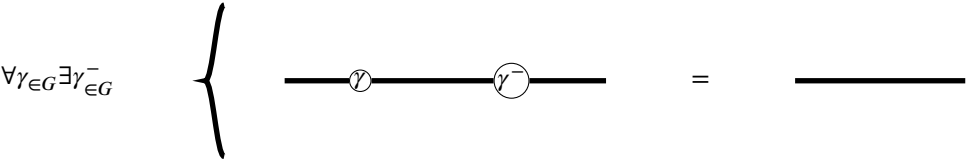
ISOMETRY

But recall that the collections of invertible transformations we are really interested in are the *rigid* ones, the ones that move objects in space without deforming them. We can identify when a transformation is rigid by the following criterion:

γ is an *isometry*
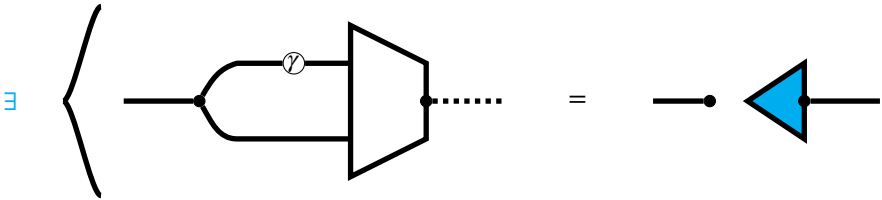
R<small>IGID DISPLACEMENTS</small>

Now we return to our sticky spiders. From now we consider sticky spiders on the open unit square, so that we can speak of shapes on a canvas. Now we will try to displace the shapes of a sticky spider. We know the planar isometries of Euclidean space can be expressed as a translation, rotation, and a bit to indicate the chirality of the shape – as mirror reflections are also an isometry.

Isometries of $\mathbf{R}^2$

$$\mathbf{x} \in \mathbf{R}^2$$

$$\theta \in S^1 \simeq [0, 2\pi)$$

$$c \in \{-1, 1\}$$

With this in mind, we have the following condition relating different spiders, telling us when one is the same as the other up to rigidly displacing shapes.

Rigid displacement

Chirality leaves us with a wrinkle: in flatland, we do not expect shapes to suddenly flip over. We would like to express just those rigid transformations that leave the chirality of the shape intact, because really we want to only be able to slide the shapes around the canvas, not leave the canvas to flip over. So we go on to define rigid continuous motion in flatland.

3.6.4   *Moving shapes*

If we want continuous transformations in the plane from the configuration of shapes in one spider to end at the configuration of shapes in another, we ought to define an analogue of *homotopy*: the continuous deformation of one map to another. However, we will have to massage the definition a little to work in our setting of continuous relations.

HOMOTOPY IN **TOPREL**

Usually, when we are restricted to speaking of topological spaces and continuous functions, a homotopy is defined:

Definition 3.6.10 (Homotopy in **Top**). where f and g are continuous maps $A \to B$, a *homotopy* $\eta : f \Rightarrow g$ is a continuous function $\eta : [0, 1] \times A \to B$ such that $\eta(0, -) = f(-)$ and $\eta(1, -) = g(1, -)$.

In other words, a homotopy is like a short film where at the beginning there is an f, which continuously deforms to end the film being a g. Directly replacing "function" with "relation" in the above definition does not quite do what we want, because we would be able to define the following "homotopy" between open sets.
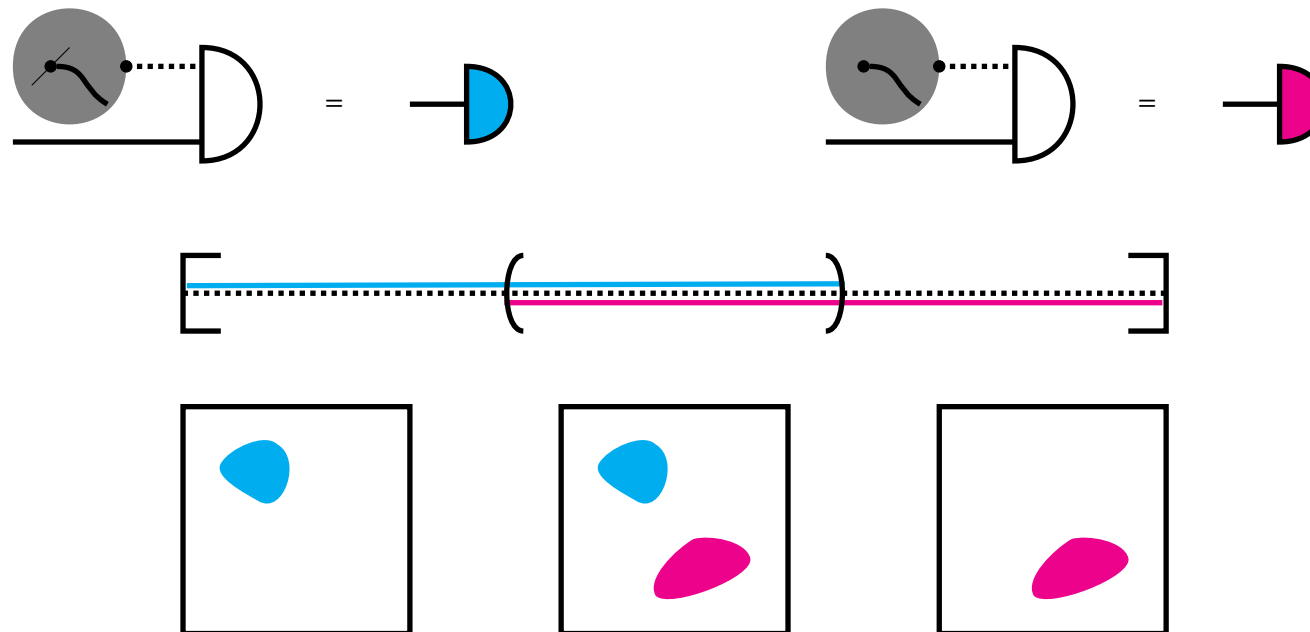
What is happening in the above film is that we have our starting open set, which stays constant for a while. Then suddenly the ending open set appears, the starting open disappears, and we are left with our ending; while *technically* there was no discontinuous jump, this isn't the notion of sliding we want. The exemplified issue is that we can patch together (by union of continuous relations) vignettes of continuous relations that are not individually total on $[0, 1]$. We can patch this problem by asking for homotopies in **TopRel** to satisfy the additional condition that they are expressible as a union of continuous partial maps that are total on the unit interval.

$$\eta(0, -) = f(-)$$

$$\eta(1, -) = g(-)$$

η is the union of homotopies of partial cts. maps

p is a partial map

p is total on $[0, 1]$
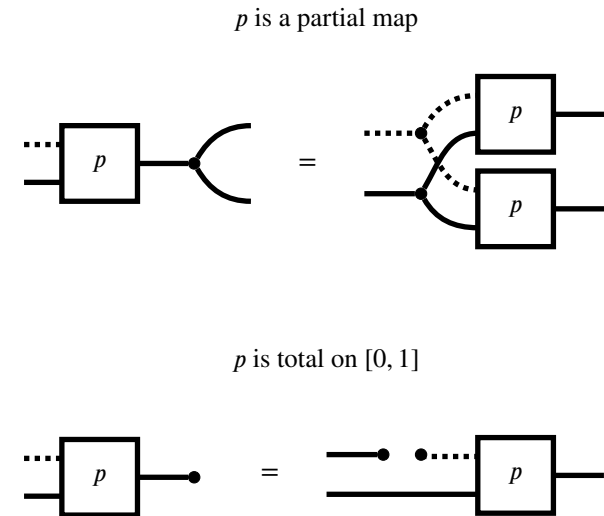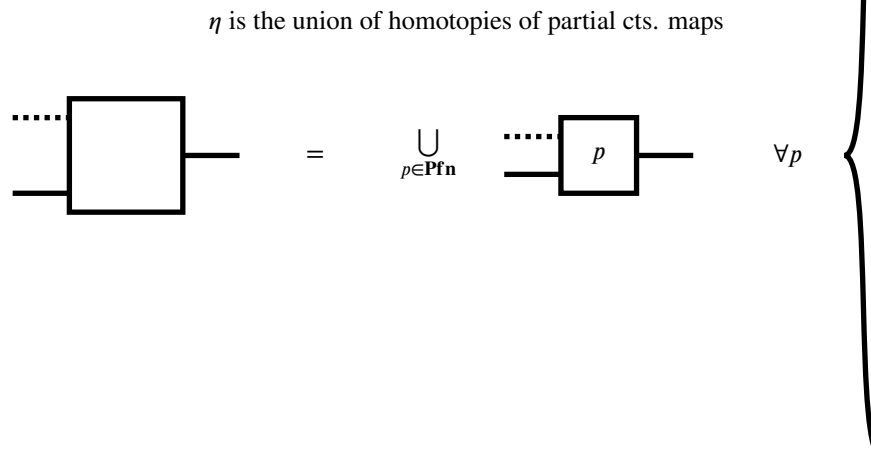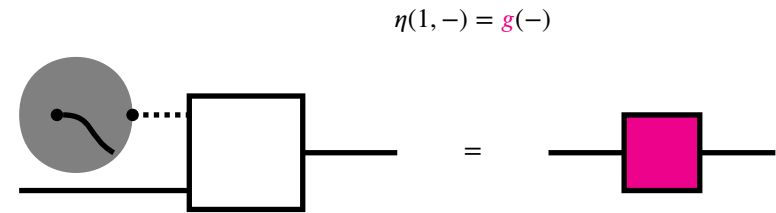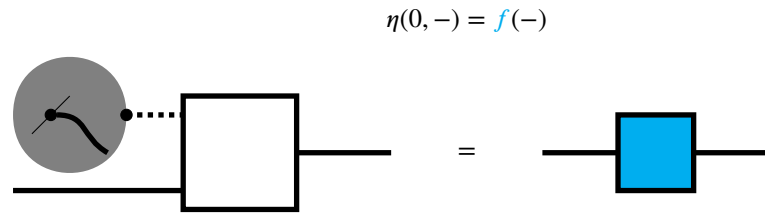
Observe that the second condition asking for decomposition in terms of partial comes for free by Proposition 3.4.20; the constraint of the definition is provided by the first condition, which is a stronger condition than just asking that the original continuous relation be total on I:
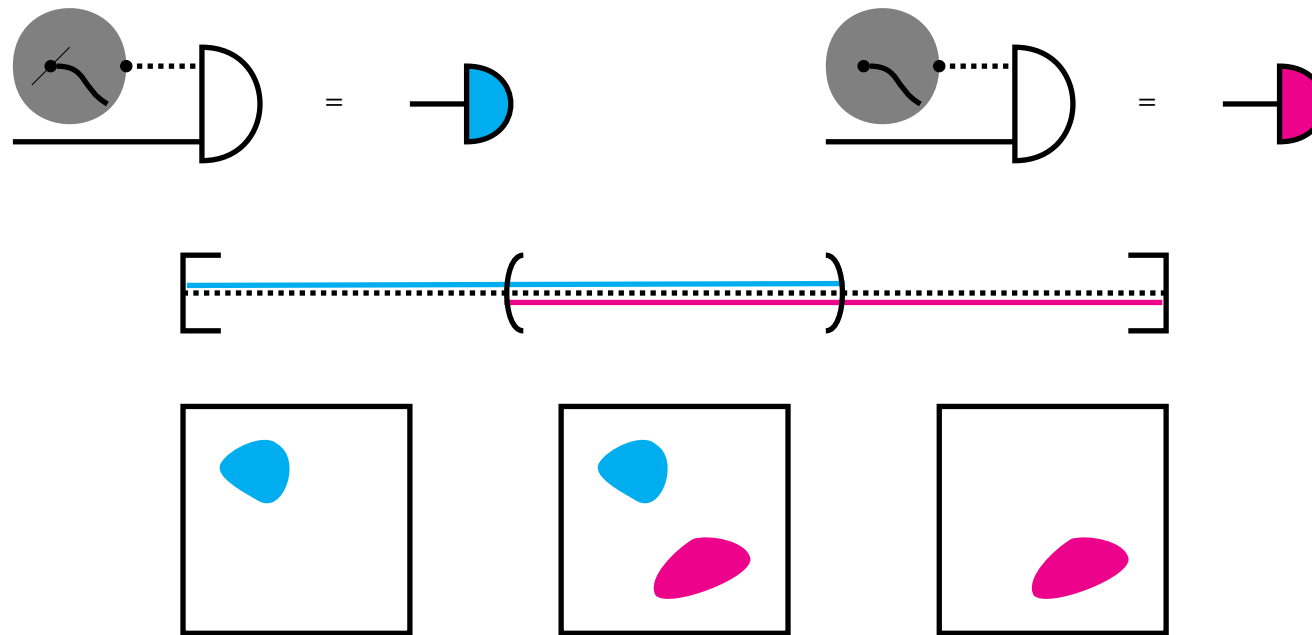
This definition is "natural" in light of Proposition 3.4.20, that the partial continuous functions $A \to B$ form a basis for **TopRel**(A, B): we are just asking that homotopies between partial continuous functions – which can be viewed as regular homotopies with domain restricted to the subspace topology induced by an open set – form a basis for homotopies between continuous relations.

(basis)   (total on $[0, 1]$)
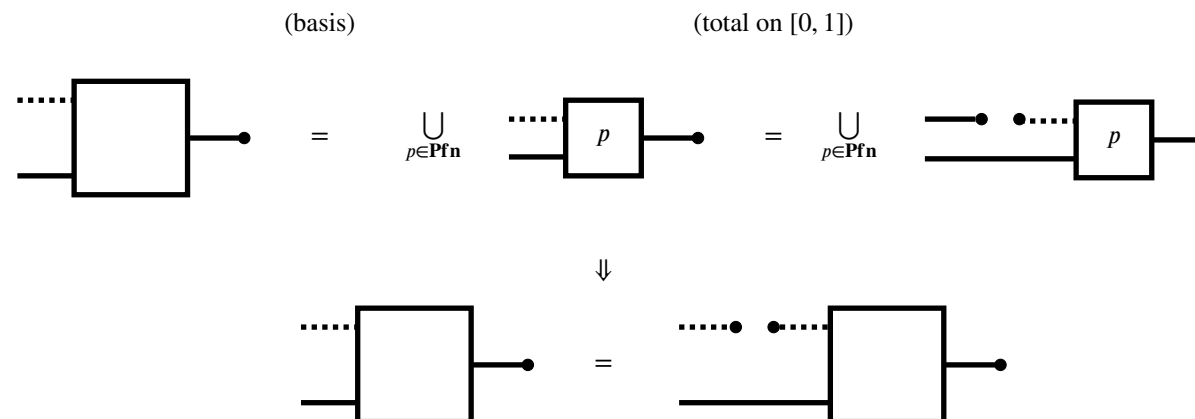
CONTRACTIBLE SPACES

With homotopies in hand, we can define a stronger notion of connected shapes with no holes, which are usually called *contractible*. The reason for the terminology reflects the method by which we can guarantee a shape in flatland has no holes: when any loop in the shape is *contractible* to a point.

V is *contractible* when:
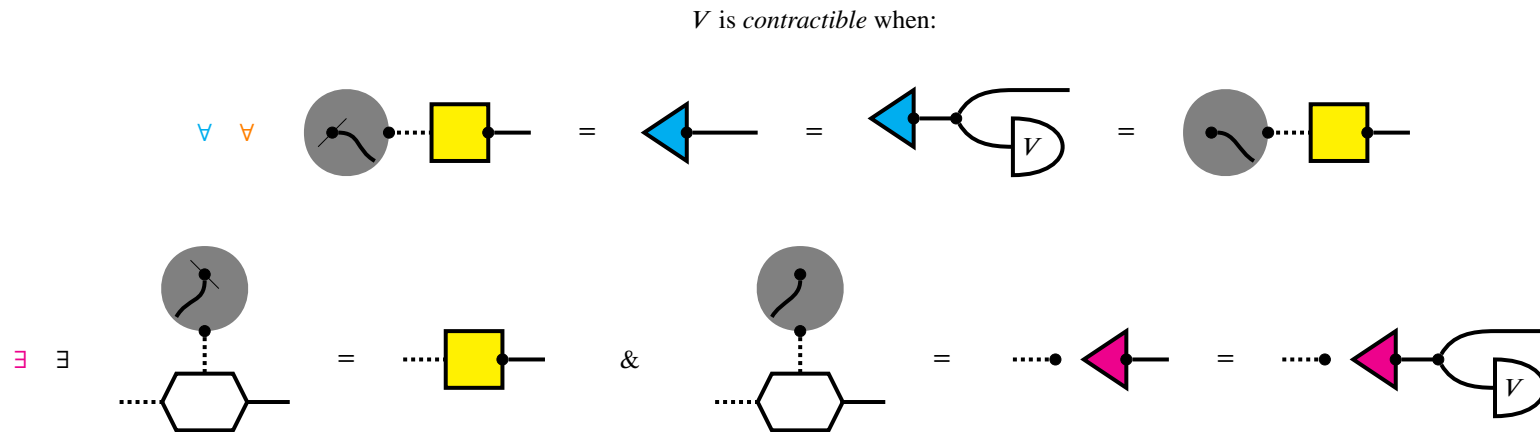
Contractible open sets are worth their own notation too; a solid black effect, this time with no hole.
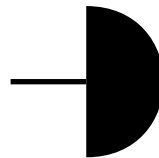
3.6.5   Rigid motion

Now at last we can define sliding shapes. What we mean by two sticky spiders being relatable by sliding shapes is that we have a homotopy that begins at one and ends at the other, such that every point in between is itself a sticky spider related to the first by rigid displacement.

Rigid motion
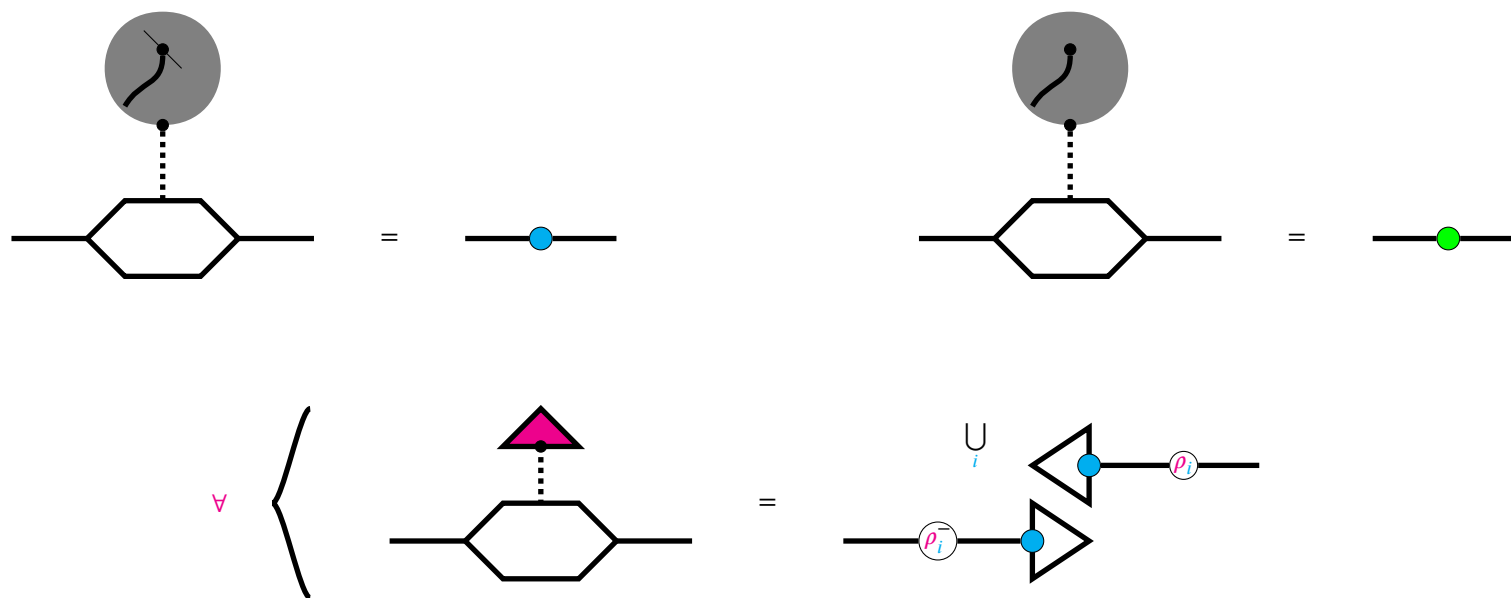
CONFIGURATION SPACES

We can depict the *configuration space* of shapes that are obtainable by displacing the shapes of a given spider by a split idempotent through the n-fold tensor of rigid transforma-
tions – a restriction to the subspace of the largest open set contained in the subset of all valid (with correct chirality) combinations of displacements that yield another spider.

Configuration space of a sticky spider

$$\overset{|i|}{\bigotimes} \mathbf{Iso(R}^2)$$

=

$$\overset{|i|}{\bigotimes} \mathbf{Iso(R}^2)$$ $$\overset{|i|}{\bigotimes} \mathbf{Iso(R}^2)$$
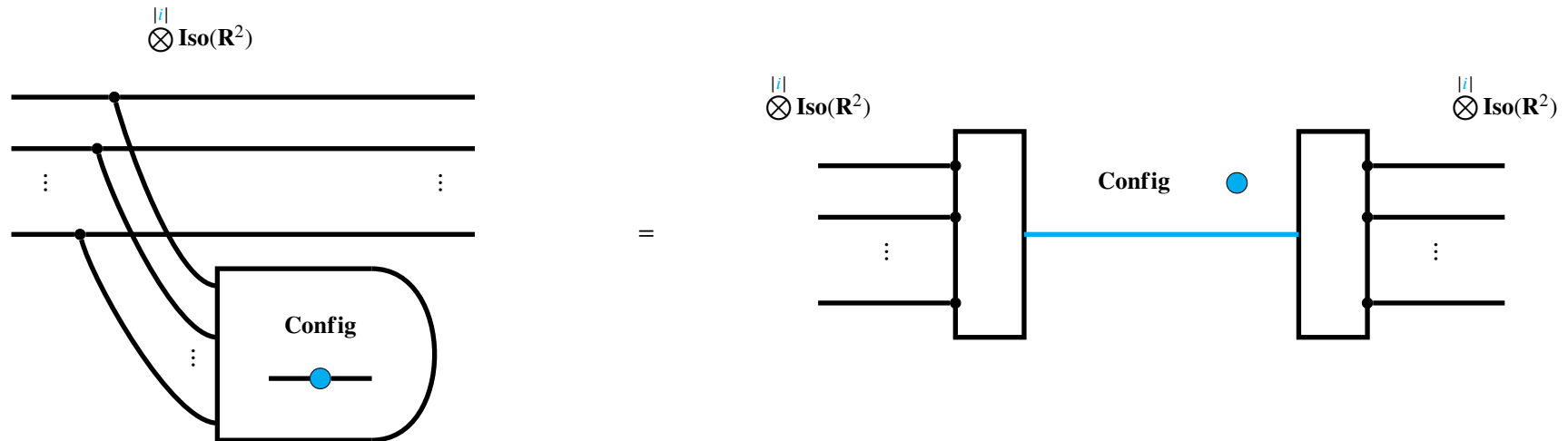
Config

Observe that the data of rigid motion on a sticky spider as we have defined above can be captured as a continuous map from the unit interval to rigid transformations: one for each shape in the spider. This is precisely a continuous path in configuration space.

$$\overset{|i|}{\bigotimes} \mathbf{Iso(R}^2)$$

=

ρ_1

ρ_i
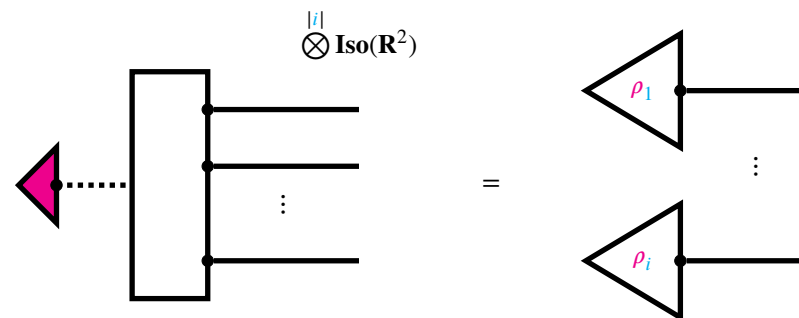
What are the connected components of configuration space? Evidently, there are pairs of spiders that are both valid displacements, but not mutually reachable by rigid motion. For example, shapes might *enclose* or *trap* other shapes, or shapes might be *interlocked*. Depicted below are some pairs of configurations that are mutually unreachable by rigid transformations:

Trapped
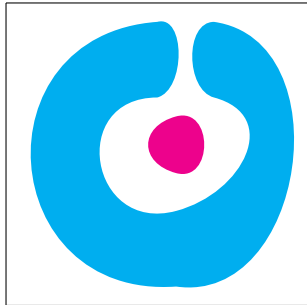
Enclosed
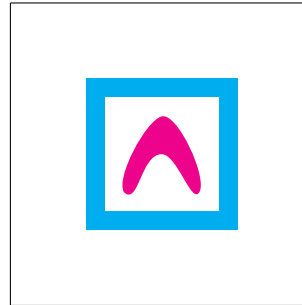
Interlocked
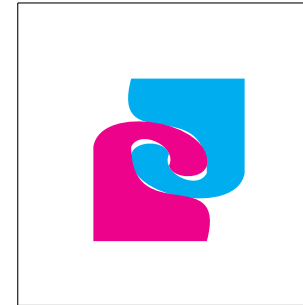
Not trapped
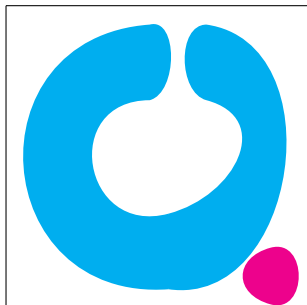
Not enclosed
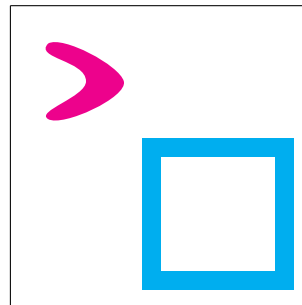
Not interlocked
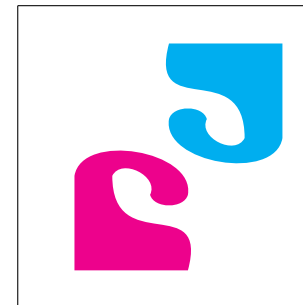
Now we have the conceptual toolkit to begin modelling these concepts in the configuration space of a sticky spider.

3.6.6   *Modelling linguistic topological concepts*

By "linguistic", I mean to refer to the kinds of concepts we use in everyday language. These are concepts that even young children have an intuitive grasp of [], but their formal definitions are difficult to pin down. One such relation modelled here – touching – is in fact a *semantic prime* []: a word that is present in essentially all natural languages that is conceptually primitive, in the sense that it resists definition in simpler terms. It is among the ranks of concepts like *wanting* or *living*, words that are understood by the experience of being human, rather than by school. As such, I make no claim that these definitions are "correct" or "canonical", just that they are good enough to build upon moving forward.

PARTHOOD

Let's say that a "part" refers to an entire simply connected component. Simply connected is already a concept in our toolkit. A shape U is disjoint from another shape V intuitively when we can cover U in a blob with no holes such that the blob has no overlap with V. So, U is a part of V when it is simply connect, wholly contained in V, and there exists a contractible open that is disjoint from V that covers U. Diagrammatically, this is:

U is *a part of V*
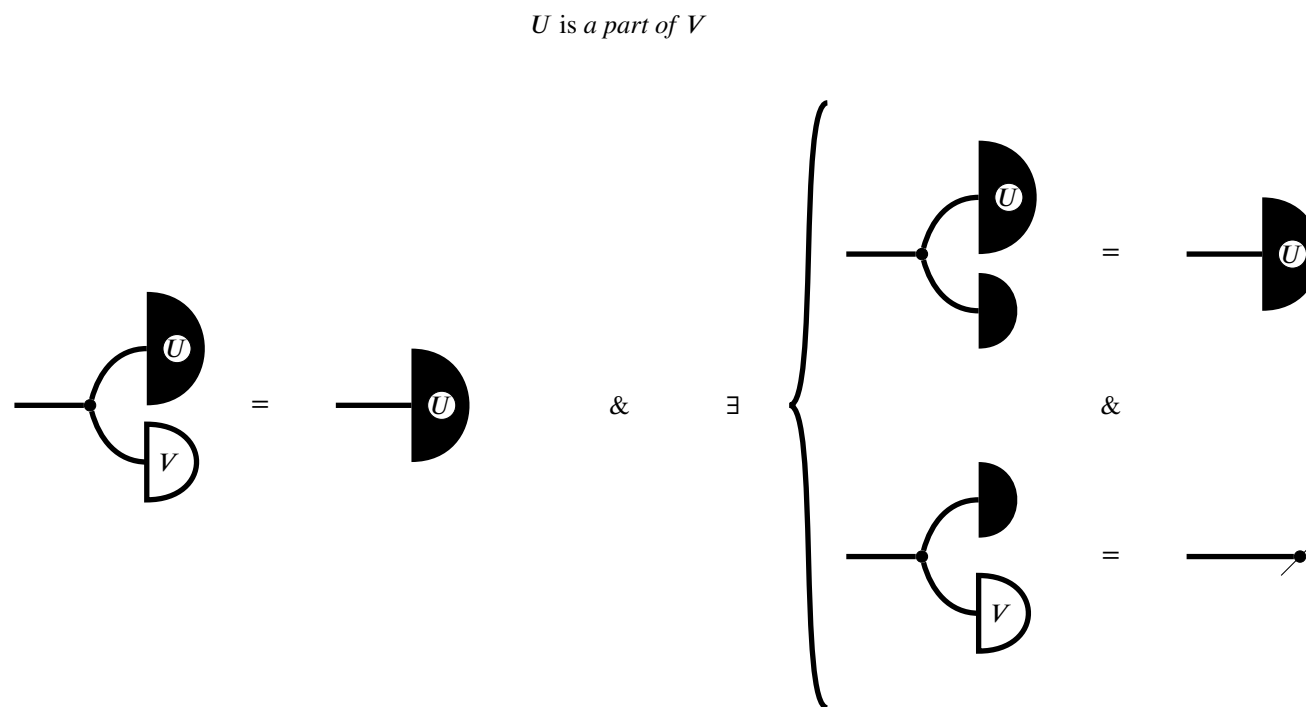
TOUCHING

Let's distinguish touching from overlap. Two shapes are "touching" intuitively when they are as close as they can be to each other, somewhere; any closer and they would overlap. Let's assume that we can restrict our attention to the parts of the shape that are touching, and that we can fill in the holes of these parts. At the point of touching, there is an infinitesimal gap – just as when we touch things in meatspace, there is a very small gap between us and the object due to the repulsive electromagnetic force between atoms. To deal with infinitesimals we borrow the $\epsilon - \delta$ trick from mathematical analysis; for any arbitrarily small δ, we can pick an even smaller ball of radius ϵ such that if we stick the ball in the gap, the ball forms a bridge that overlaps the two filled-in shapes, which allows us to draw a continuous line between them. Diagrammatically, this is:

U and V are *touching*

WITHIN

If U surrounds V, or equivalently, if V is within U, then we are saying that leaving V in almost any direction, we will see some of U before we go off to infinity. We can once again use open balls for this purpose, which correspond to possible places you can get to from a starting point **x** within a distance ϵ. In prose, we are asking that any open ball that contains all of U must also contain all of V.

$$V \text{ is } within \text{ } U, \text{ or } U \text{ } surrounds \text{ } V$$
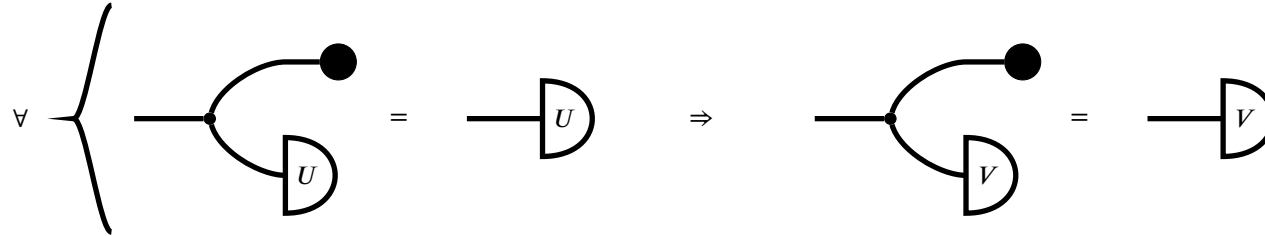
CONTAINERS AND ENCLOSURE

There is a strong version of within-ness, which we will call enclosure. As in when we are in elevators and the door is shut, nothing gets in or out of the container. Intuitively, there is a hole in the container surrounded on all sides, and the contained shape lives within the hole. To give a real-world example, honey lives within a honeycomb cell in a beehive, but whether the honey is enclosed in the cell depends on whether it is sealed off from air with beeswax. So in prose we are asking that any way we fill in the holes of the container with a blob, that blob must cover the contained shape. Diagrammatically, this amounts to levelling up from open balls in our previous definition to contractible sets:

$$U \text{ } encloses \text{ } V$$
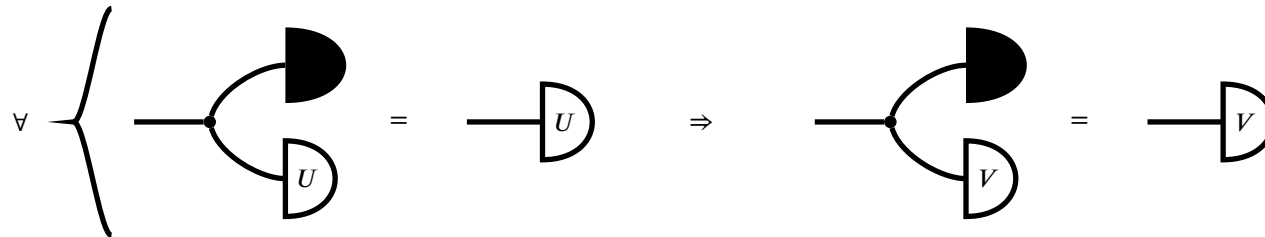
TRAPPED

There is an intermediate notion between within-ness and enclosure; for instance, standing in the stonehenge you are surrounded by the pillars, but you can always walk away, whereas if the pillars are very close, such as the bars of a jail cell, a human would not be able to leave the trap while still being able to see the outside. The difficulty here is that relative sizes come into play: small animals would still consider it a case of mere within-ness, because they can still walk away between the bars. So we would like to say that no matter how the pair of objects move rigidly, being trapped means that the trapped V stays within U. In other words, that in configuration space, if we forget about all other shapes, we can partition our space of configurations by two concepts, whether V is within U or not, and moreover that these two components is disjoint – i.e. not simply connected – so there is no rigid motion that can allow V to escape from being within U if V starts off trapped inside in U.
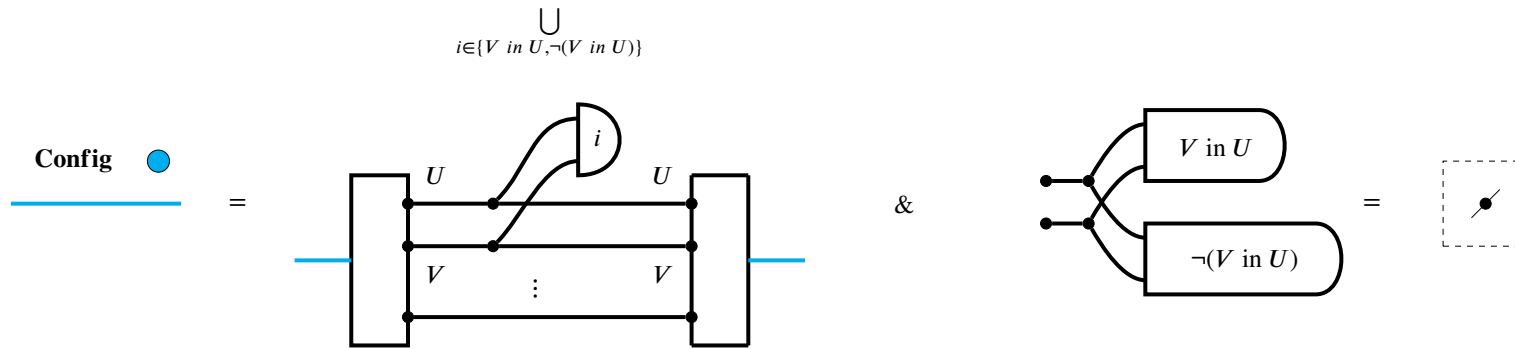
INTERLOCKED

Two shapes might be tightly interlocked without being inside one another. Some potentially familiar examples are plastic models of molecular structure that we encounter in school, metal lids in cold weather that are too tightly hugging the glass jar, or stubborn Lego pieces that refuse to come apart. The commonality of all these cases is that the two shapes must move together as one, unless deformed or broken. In other words, when two shapes are interlocked, knowing the position in space of one shape determines the position of the other, and this determination is a fixed isometry of space. So we only need to specify a range of positions S for the entire subconfiguration of interlocked shapes U and V, and we may obtain their respective positions by a fixed rigid motion ρ. Since objects may interlock in multiple ways, we may have a sum of these expressions. We additionally observe that interlocking shapes should also be touching, which translates to containment inside the touching concept. Finally, we observe that as in the case of entrapment and enclosure, rigid motions are interlocking-invariant, which translates diagrammatically to the constraint that each S, ρ expression is an entire connected component in configuration space.
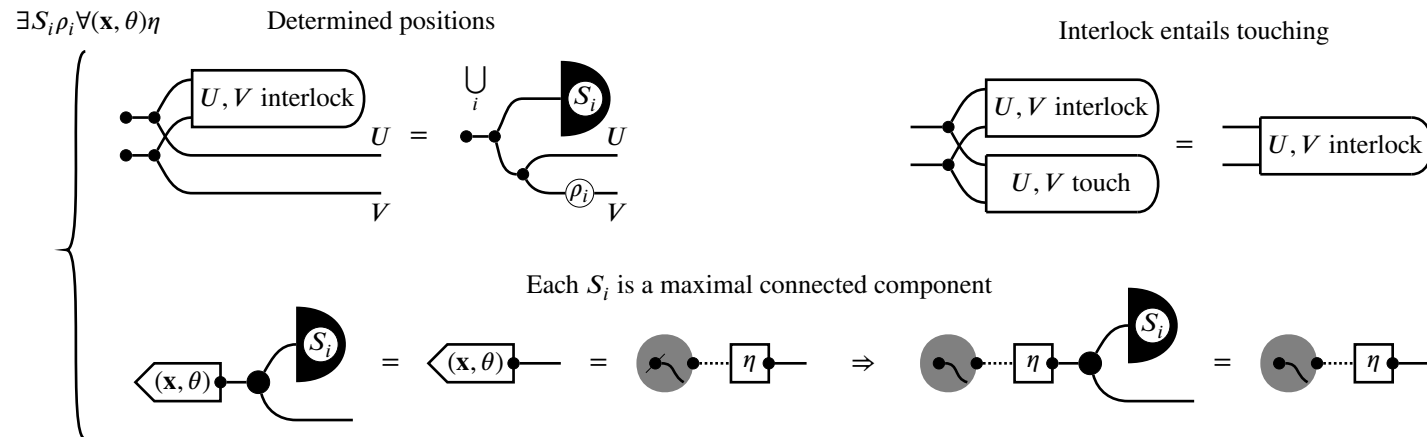
CONSTRAINED MOTION

A weaker notion of interlocking is when shapes only imperfectly determine each other's potential displacements, by specifying an allowed range. Here is an understatement: there is some interest in studying how shapes mutually constrain each other's movements in this way.

Chain (loose)   Chain (taut)   Gears   Switch
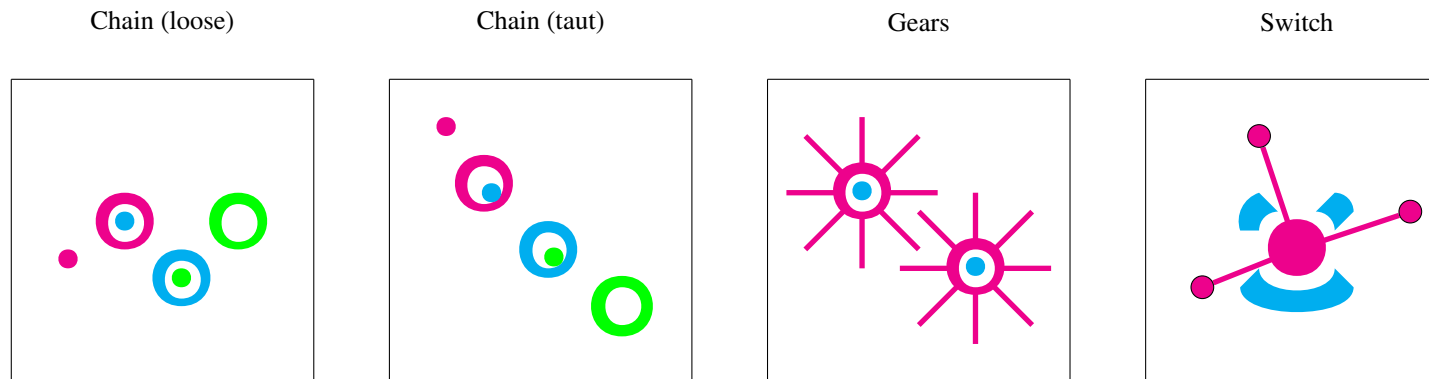
There are as many definitions to go through here as there are potential mechanical models, and among other things, there are mechanically realised clocks [], computers [], and analogues of electric circuits []. So instead, we will allow ourselves to additionally specify open sets as concepts in configuration space that correspond to whatever mechanical concepts we please, and we assure the reader seeking rigour that blueprints exist for all the mechanisms humans have built. Of course in reality mechanical motions are reversible among rigid objects, and directional behaviour is provided by a source of energy, such as gravitational potential, or wound springs. But we may in principle replace these sources of energy by a belt that we choose to spin in one direction – our own arrow of time. We postpone discussion of causal-mechanistic understanding and analogy for a later section.

3.6.7 States, actions, manner

Configuration space explains why we label noun wires: each wire in expanded configuration space must be labelled with the shape within the sticky spider it corresponds to so that the section and retract know how to reconstruct the shapes, since each shape may have a different spatial extent.

U V

$U^{\bullet}$

$V^{\bullet}$
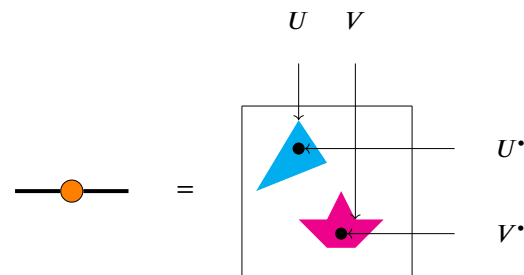
Concretely, for each shape in a sticky spider,

in order to reconstruct the shape,

the data of configuration space

only has to remember a basepoint

(which the isometries act upon)

paired with a label naming the shape

(so that the extent of the shape is reconstructible)

$(U, U^{\bullet} \in \mathbf{Iso}(\mathbb{R}^2))$

Config

$(V, V^{\bullet} \in \mathbf{Iso}(\mathbb{R}^2))$

Config =

$(U, \mathbf{Iso}(\mathbb{R}^2))$

$(V, \mathbf{Iso}(\mathbb{R}^2))$

$(U, \mathbf{Iso}(\mathbb{R}^2))$

$(V, \mathbf{Iso}(\mathbb{R}^2))$
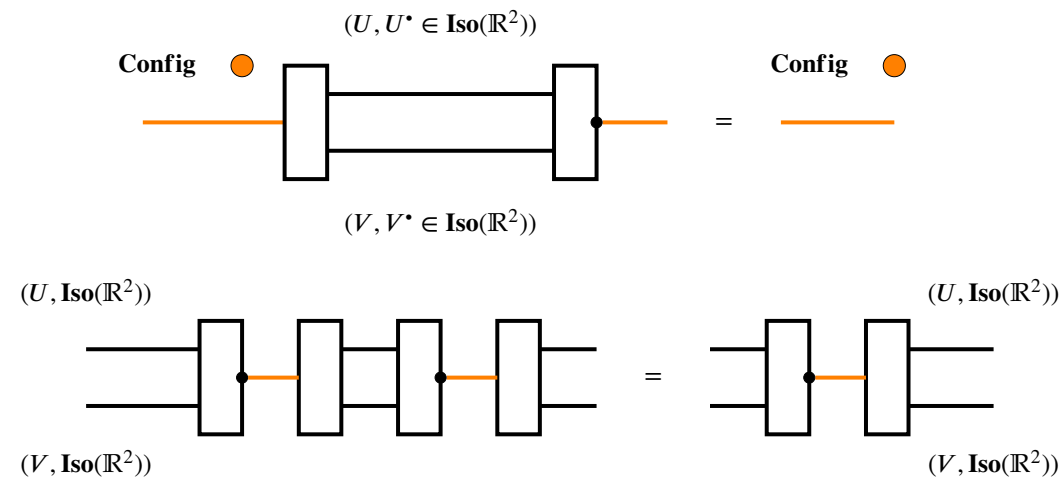
All of the concepts we have defined so far are open sets in configuration space – and for any concept that isn't, we are always free to take the interior of the set; the largest open set contained within the concept. Passing through the split idempotent, we can recast each as a circuit gate using copy maps.
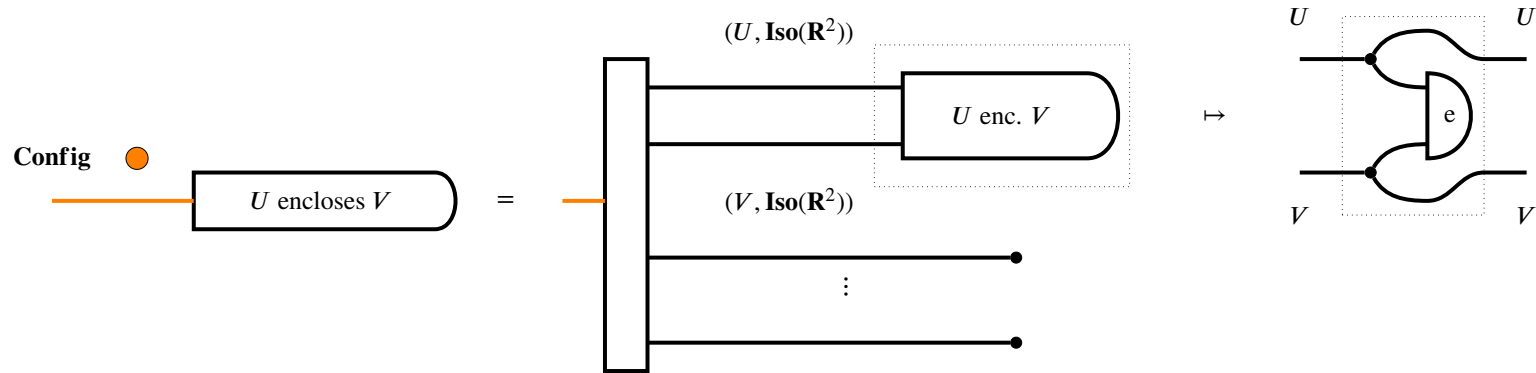
Going forward, we will just label the wires with the names of each shape when necessary. We notice that one feature of this procedure to get gates from open sets is that all gates commute, due to the commutativity of copy.
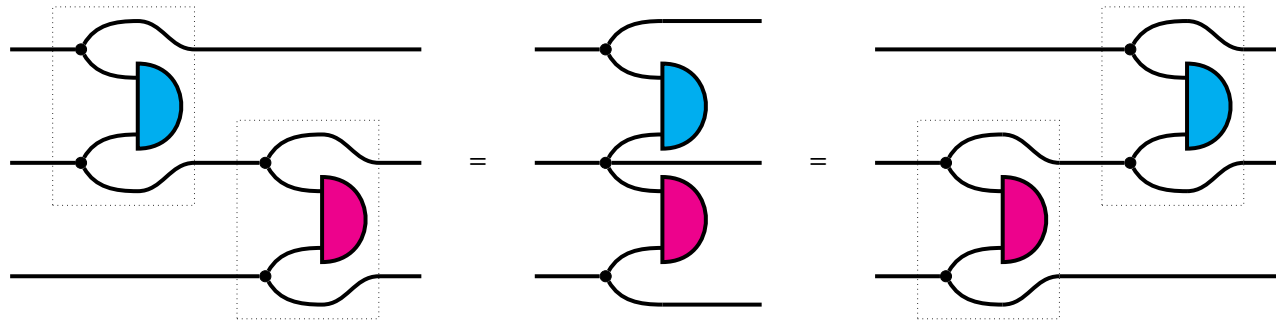
Moreover, since each gate of this form is a restriction to an open set, the gates are idempotent. So the concepts we have defined so far behave as if describing *states* of affairs in space, as if we adding commuting adjectives to space to elaborate detail. For example, `fast red car`, `fast car that is red`, `car is (red and fast)` all mean the same thing. As we add on progressively more concepts, we get diminishing subspaces of configurations in the intersection of all the concepts. So the natural extension is to ask how states of affairs can change with motion. A simple example is the case of *collision*, where two shapes start off not touching, and then they move rigidly towards one another to end up

touching.

A particular collision trajectory

At $t = 0$, the shapes do not touch   At $t = 1$, the shapes touch
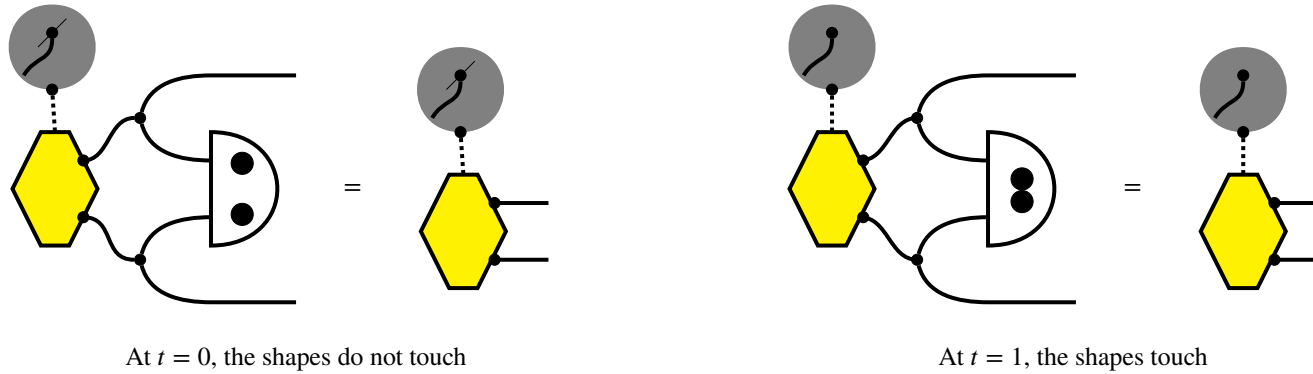
Recalling that homotopies between relations are the unions of homotopies between maps, we have a homotopy that is the union of all collision trajectories, which we mark ∀. Now we seek to define the interior $i(\forall)$ as the concept of collision; the expressible collection of all particular collisions. But this is not just an open set on the potential configuration of shapes, it is a collection of open sets parameterised by homotopy.
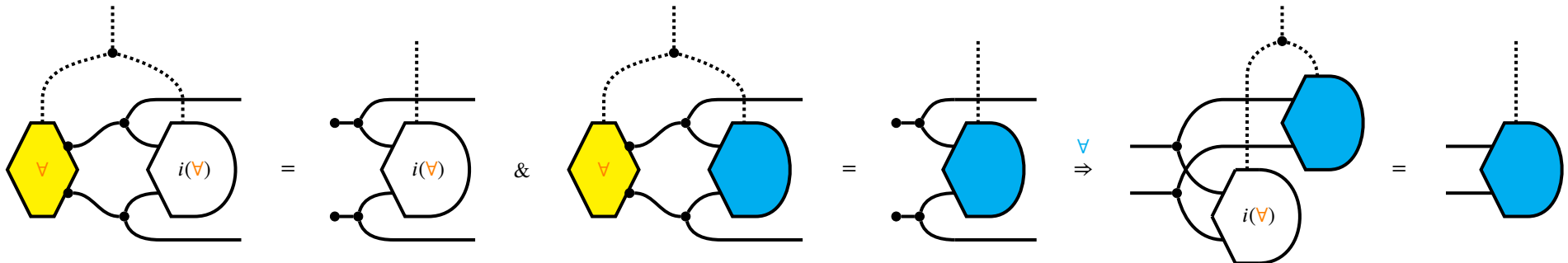
Once we have the open set $i(\forall)$ that corresponds to all expressible collisions, we have a homotopy-parameterised gate. Following a similar procedure, we can construct gates of motion that satisfy whatever pre- and post-conditions we like.
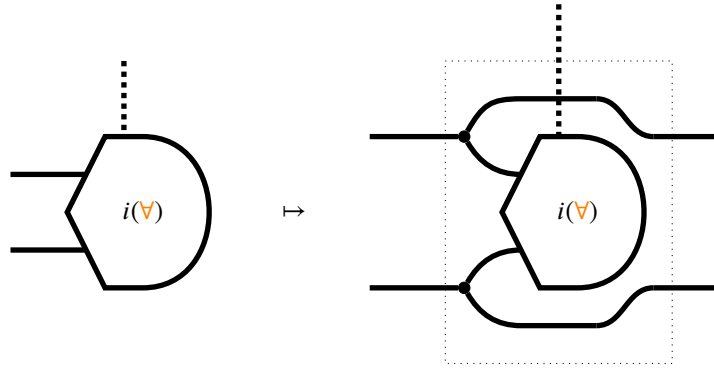
We can compose multiple rigid motions sequentially by a continuous function ; that splits a single unit interval into two: ; $:= x \mapsto \begin{cases} (2x, 0) \text{ if } x \in [0, \frac{1}{2}) \\ (1, 2x - 1) \text{ if } x \in [\frac{1}{2}, 1] \end{cases}$. The effect of the map is to splice two vignettes of the same length together by doubling their speed, then placing them one after the other. We can achieve the same thing without resorting to units of measurement, because recall by Theorem 3.6.7 and by construction that we have access to a map that selects midpoints for us; we will revisit a string-diagrammatic treatment of homotopy and tenses in a later section. We can also compose multiple motions in parallel by copying the unit interval, allowing it to parameterise multiple gates simultaneously.

Sequential composition of motions   Parallel composition of motions
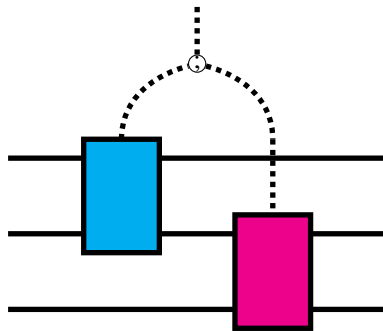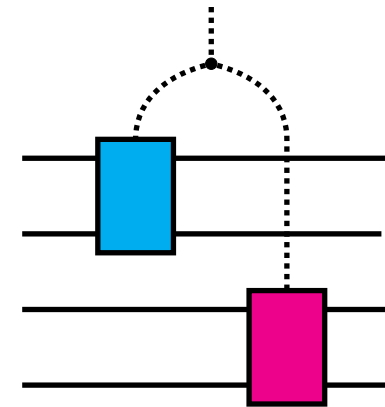
It is easy to see that the gates can always be rewritten to respect the composition order given by ; and copy, since for any input point at the unit interval the gates behave as restrictions to open sets. These new gates do not generally commute; consider comparing the situation where a tenant moves into one apartment and then another, with the situation where

the tenant reverses the order of the apartments. These are different paths, as the postconditions must be different. So now we have noncommuting gates that model *actions*, or verbs. What kinds of actions are there? In our toy setting, in general we can define actions that arbitrarily change states of affairs if we do not restrict ourselves to rigid motions. The trick to doing this is the observation that arbitrary homotopies allow deformations, so our verb gates allow shapes to shrink and open and bend in the process of a homotopy, as long as at the end they arrive at a rigid displacement of their original form.

Open   Squeeze   Leave then stretch / Move   Shut / Rotate

Enclosed   Not enclosed
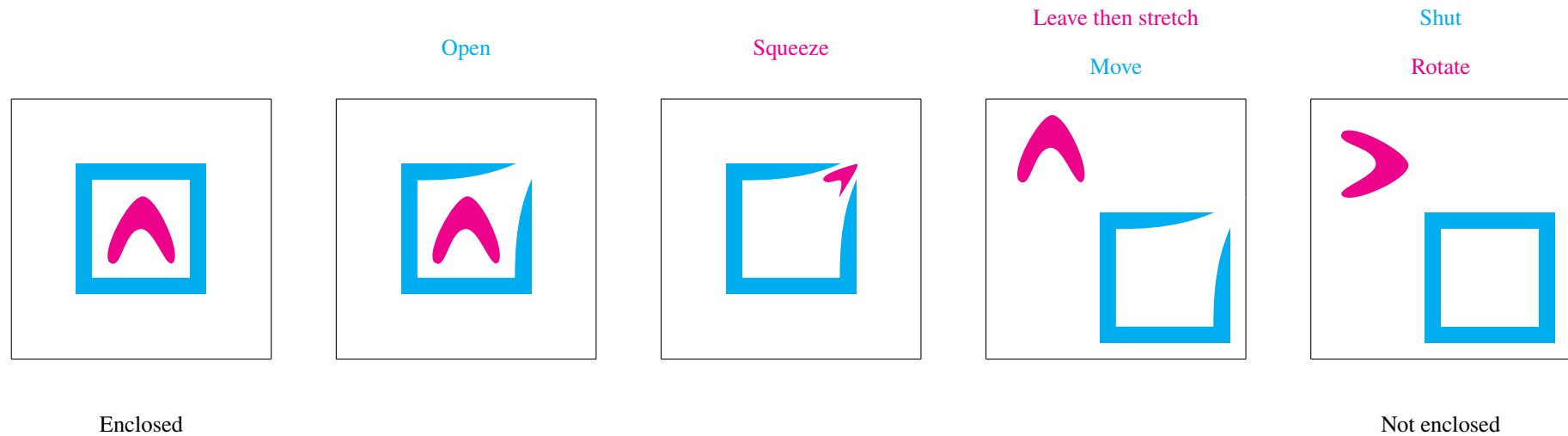
We can further generalise by noting that completely different spiders can be related by homotopy, so we can model a situation where there is a permanent bend, or how a rigid shape might shatter.

Break   Melt   Same shape / Same shape

Interlocked   Not interlocked
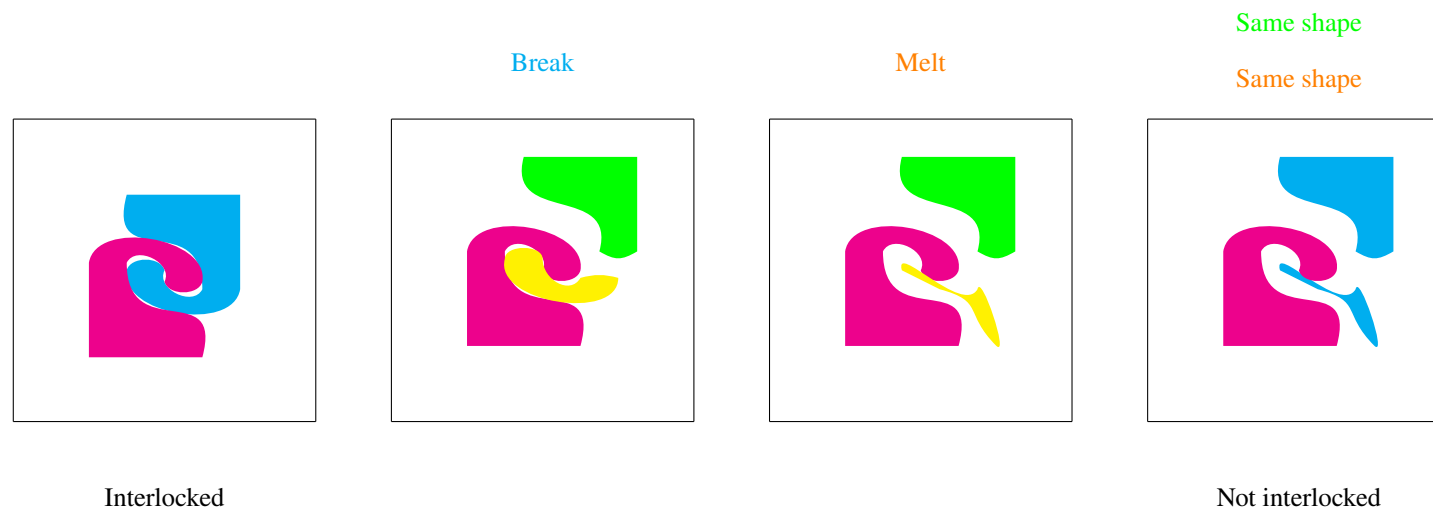
We provide the following construction as a general recipe to construct homotopies between spiders.

Construction 3.6.11 (Morphing sticky spiders with homotopies). We aim to construct homotopies relating (almost) arbitrary sticky spiders. For now we focus on just changing one shape into another arbitrary one. The idea is as follows. First, we need a cover of open balls $\cup \mathcal{J} = T^0$ and $\cup \mathcal{K} = T^1$ of the start and end cores T^0 and T^1 such that each $k \in T^1$ is expressible as a rigid isometry of some core $j \in \mathcal{J}$; this is so we can slide and rearrange open balls comprising T^0 and reconstruct them as T^1. As an intermediate step to eliminate holes and unify connected components, we gather all of the balls at a meeting point m (to be determined shortly.) Intuitively we can illustrate this process as follows:

Open balls cover core

collapse halo to m

centre ball cover

m

rendezvous m in overlap
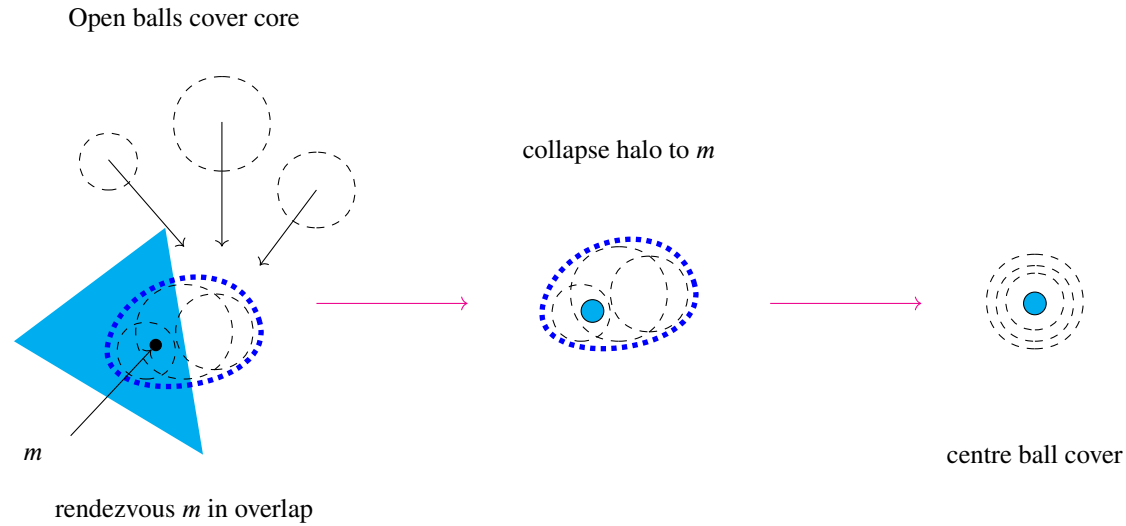
Second, in order to perform the sliding of open balls, we observe that, given a basepoint to act as origin (which we assume is provided by the data of the split idempotent of configuration space) we can express the group action of rigid isometries $\mathbf{Iso}(\mathbf{R}^2)$ on $\mathbf{R}^2$ as a continuous function:

$$((\mathbf{a}, \theta), \mathbf{b}) \mapsto \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \mathbf{b} + \mathbf{a}$$

Third, before we begin sliding the open balls, we must ensure that the halo of the shape cooperates. We observe that a given shape i in a sticky spider may be expressed as the union of a family of constant continuous partial functions in the following way. Given an open cover $\mathcal{J}$ such that $\cup \mathcal{J} = T_i$, where T_i is the core of the shape i, each function is a constant map from some $T_j \in \mathcal{J}$ to some point $x \in S_i$, where S_i is the halo of the shape i. For each $T_j \in \mathcal{J}$ and every point $x \in S_i$, the constant partial function that maps T_j to x is in the

family.

$$\bigcup_{T_j \in \mathcal{J}_i \subseteq \tau \,:\, \cup \mathcal{J}_i = T_i}$$

$$\tikz{} \quad T_i \quad S_i \quad = \quad T_j \quad x \quad \bigcup_{x \in S_i}$$

By definition of sticky spiders, there must exist some point m that is in both the core and the halo: we pick such a point as the rendezvous for the open balls. For each partial map in the family, we provide a homotopy that varies only the image point x continuously in the space to finish at m. Now we can slide the open balls to the rendezvous m. Since homotopies are reversible by the continuous map $t \mapsto (1-t)$ on the interval, we can perform the above steps for shapes T^0 and T^1 to finish at the same open ball, reversing the process for T^1 and composing sequentially to obtain a finished transformation. The final wrinkle to address is when dealing with multiple shapes. Recalling our exclusion conditions **??** for shapes, it may be that parts of one shape are enclosed in another, so the processes must be coordinated so that there are no overlaps. For example, the enclosing shape must be first opened, so that the enclosed shape may leave. I will keep it an article of faith that such coordinations exist. I struggle to come up with a proof that all spiders $\mathbf{R}^2$ are mutually transformable by homotopy in this (or any other) way, so that will remain a conjecture. But it is clear that a great deal of spiders are mutually transformable; almost certainly any we would care to draw. So this will just be a construction for now.

3.7   Mathematician's endnotes

This section has two aims. First is to formally demonstrate that **TopRel** is indeed a symmetric monoidal category. Second is to investigate the relationship between **TopRel** and what seem like they should be close cousins: **Rel**, **Top**, and **Loc**. We demonstrate that **TopRel** enjoys a free-forgetful adjunction with **Rel** as expected, but **TopRel** has no forgetful functor to **Loc**. We verify that **TopRel** cannot be viewed as "powering up topology with relations" in the usual ways. Specifically, **TopRel** does not arise as conservative generalisation of the Kleisli category of the powerset monad on **Set** to **Top**, nor is it equivalent to **Span**(**Top**). We provide a sketch involving display categories to attempt to explain where the topology is coming from. The failure of these (relatively sophisticated and general) techniques to modify **Top** to accommodate relations may explain why **TopRel** has no footprint in the literature, and suggests that the study of this category may be a novel contribution.

3.7.1   The category *TopRel*

Proposition 3.7.1 (**TopRel** is a category)**.** continuous relations form a category **TopRel**.

Proof. IDENTITIES: Identity relations, which are always continuous since the preimage of an open U is itself.

COMPOSITION: The normal composition of relations. We verify that the composite $X^\tau \xrightarrow{R} Y^\sigma \xrightarrow{S} Z^\theta$ of continuous relations is again continuous as follows:

$$U \in \theta \implies S^\dagger(U) \in \sigma \implies R^\dagger \circ S^\dagger(U) = (S \circ R)^\dagger \in \tau$$

ASSOCIATIVITY OF COMPOSITION: Inherited from **Rel**.   □

3.7.2   Symmetric Monoidal structure

Proposition 3.7.2. (**TopRel**, •, $X^\tau \otimes Y^\sigma := (X \times Y)^{(\tau \times \sigma)}$) is a symmetric monoidal closed category.

Proof. TENSOR UNIT: The one-point space •. Explicitly, $\{\star\}$ with topology $\{\varnothing, \{\star\}\}$.

TENSOR PRODUCT: For objects, $X^\tau \otimes Y^\sigma$ has base set $X \times Y$ equipped with the product topology $\tau \times \sigma$. For morphisms, $R \otimes S$ the product of relations. We show that the tensor of continuous relations is again a continuous relation. Take continuous relations $R : X^\tau \to Y^\sigma, S : A^\alpha \to B^\beta$, and let U be open in the product topology ($\sigma \times \beta$). We need to prove that $(R \times S)^\dagger(U) \in (\tau \times \alpha)$. We may express U as $\bigcup_{i \in I} y_i \times b_i$, where the y_i and b_i are in the bases $\mathfrak{b}_\sigma$ and $\mathfrak{b}_\beta$ respectively. Since for any relations we have that

$R(A \cup B) = R(A) \cup R(B)$ and $(R \times S)^\dagger = R^\dagger \times S^\dagger$:

$$(R \times S)^\dagger (\bigcup_{i \in I} y_i \times b_i)$$

$$= \bigcup_{i \in I} (R \times S)^\dagger (y_i \times b_i)$$

$$= \bigcup_{i \in I} (R^\dagger \times S^\dagger)(y_i \times b_i)$$

Since each y_i is open and R is continuous, $R^\dagger(y_i) \in \tau$. Symmetrically, $S^\dagger(b_i) \in \alpha$. So each $(R^\dagger \times S^\dagger)(y_i \times b_i) \in (\tau \times \alpha)$. Topologies are closed under arbitrary union, so we are done.

THE NATURAL ISOMORPHISMS ARE INHERITED FROM **REL**. We will be explicit with the unitor, but for the rest, we will only check that the usual isomorphisms from **Rel** are continuous in **TopRel**. To avoid bracket-glut, we will vertically stack some tensored expressions.

UNITORS: The left unitors are defined as the relations $\lambda_{X^\tau} : \bullet \times X^\tau \to X^\tau := \{ (\begin{pmatrix} \star \\ x \end{pmatrix}, x) \mid x \in X \}$, and we reverse the pairs to obtain the inverse $\lambda_{X^\tau}^{-1}$. These relations are continuous since the product topology of τ with the singleton is homeomorphic to τ: $U \in \tau \iff (\bullet, U) \in (\bullet \times \tau)$. These relations are evidently inverses that compose to the identity. The construction is symmetric for the right unitors ρ_{X^τ}.

ASSOCIATORS: The associators $\alpha_{X^\tau Y^\sigma Z^\rho} : ((X \times Y) \times Z)^{((\tau \times \sigma) \times \rho)} \to (X \times (Y \times Z))^{(\tau \times (\sigma \times \rho))}$ are inherited from **Rel**. They are:

$$\alpha_{X^\tau Y^\sigma Z^\rho} := \{ ((\begin{pmatrix} x \\ y \end{pmatrix}, z), (x, \begin{pmatrix} y \\ z \end{pmatrix})) \mid x \in X, \, y \in Y, \, z \in Z \}$$

To check the continuity of the associator, observe that product topologies are isomorphic in **Top** up to bracketing, and these isomorphisms are inherited by **TopRel**. The inverse of the associator has the pairs of the relation reversed and is evidently an inverse that composes to the identity.

BRAIDS: The braidings $\theta_{X^\tau Y^\sigma} : (X \times Y)^{\tau \times \sigma} \to (Y \times X)^{\sigma \times \tau}$ are defined:

$$\{ (\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} y \\ x \end{pmatrix}) \mid x \in X, \, y \in Y \}$$

The braidings inherit continuity from the isomorphisms between $X^\tau \times Y^\sigma$ and $Y^\sigma \times X^\tau$ in **Top**. They inherit everything else from **Rel**

COHERENCES: Since we have verified all of the natural isomorphisms are continuous, it suffices to say that the coherences [] are inherited from the symmetric monoidal structure of **Rel** up to marking objects with topologies. □

MONOIDAL CLOSURE: Here is the evaluator.

$$placeholder$$

3.7.3   Rig category structure

Definition 3.7.3 (Biproducts and zero objects)**.** A *biproduct* is simultaneously a categorical product and coproduct. A *zero object* is both an initial and a terminal object. **Rel** has biproducts (the coproduct of sets equipped with reversible injections) and a zero object (the empty set).

Proposition 3.7.4. **TopRel** has a zero object.

Proof. As in **Rel**, there is a unique relation from every object to and from the empty set with the empty topology.   □

Proposition 3.7.5. **TopRel** has biproducts.

Proof. The biproduct of topologies X^τ and Y^σ is their direct sum topology $(X \sqcup Y)^{(\tau+\sigma)}$ – the coarsest topology that contains the disjoint union $\tau \sqcup \sigma$. As in **Rel**, the (in/pro)jections are partial identities, which are continuous by construction. To verify that it is a coproduct, given continuous relations $R : X^\tau \to Z^\rho$ and $S : Y^\sigma \to Z^\rho$, where the disjoint union $X \sqcup Y$ of sets is $\{x_1 \mid x \in X\} \cup \{y_2 \mid y \in Y\}$, we observe that $R + S := \{(x_1, z) \mid (x, z) \in R\} \cup \{(y_2, z) \mid y \in S\}$ is continuous and commutes with the injections as required. The argument that it is a product is symmetric.   □

Remark 3.7.6. Biproducts yield another symmetric monoidal structure which the $\times$ monoidal product distributes over appropriately to yield a rig category. Throughout the chapter we have been using $\cup$, but we could have also "diagrammatised" $\cup$ by treating it as a monoid internal to **TopRel** viewed as a symmetric monoidal category with respect to the biproduct. There are two diagrammatic formalisms for rig categories that we could have used, [] and []. Neither case is perfectly suitable due to the fact that we sometimes took unions over arbitrary indexing sets, which is alright in topology but not depictable as a finite diagram in the $\oplus$-structure. A neat fact that follows is that a topological space is compact precisely when any arbitrarily indexed $\cup$ of tests in the $\times$-structure is *depictable* in the $\oplus$-structure of either diagrammatic calculus for rig categories. **FdHilb** also has a monoidal product notated $\otimes$ that distributes over the monoidal structure given by biproducts $\oplus$. In contrast, we have used $\times$ – the cartesian product notation – for the monoidal product of **TopRel** since that is closer to what is familiar for sets.

3.7.4   *TopRel* and *Rel* are related by a free-forgetful adjunction

We provide free-forgetful adjunctions relating **TopRel** to **Rel** by "forgetting topology" and sending sets to "free" discrete topologies.

WE EXHIBIT A FREE-FORGETFUL ADJUNCTION BETWEEN **REL** AND **TOPREL**.

Lemma 3.7.7 (Any relation R between discrete topologies is continuous)**.** *Proof.* All subsets in a discrete topologies are open.   □

Definition 3.7.8 (L: **Rel** → **TopRel**)**.** We define the action of the functor L:

On objects $L(X) := X^\star$, (X with the discrete topology)

On morphisms $L(X \xrightarrow{R} Y) := X^\star \xrightarrow{R} Y^\star$, the existence of which in **TopRel** is provided by Lemma 3.7.7.

Evidently identities and associativity of composition are preserved.

Definition 3.7.9 (R: **TopRel** → **Rel**). We define the action of the functor R as forgetting the topological structure.

On objects $R(X^\tau) := X$

On morphisms $R(X^\tau \xrightarrow{S} Y^\sigma) := X \xrightarrow{S} Y$

Evidently identities and associativity of composition are preserved.

Lemma 3.7.10 ($RL = 1_{\mathbf{Rel}}$). The composite RL (first L, then R) is precisely equal to the identity functor on **Rel**.

Proof. On objects, $FU(X) = F(X^\star) = X$. On morphisms, $FU(X \xrightarrow{R} Y) = F(X^\star \xrightarrow{R} Y^\star) = X \xrightarrow{R} Y$ $\square$

Reminder 3.7.11 (Coarser and finer). Given a set of points X with two topologies X^τ and X^σ, if $\tau \subset \sigma$, we say that τ *is coarser than* σ, or σ *is finer than* τ.

Lemma 3.7.12 (Coarsening is a continuous relation). Let X^σ be coarser than X^τ. The identity relation on underlying points $X^\tau \xrightarrow{1_X} X^\sigma$ is then a continuous relation.

Proof. The preimage of the identity of any open set $U \in \sigma, U \subseteq X$ is again U. By definition of coarseness, $U \in \tau$. $\square$

Proposition 3.7.13 ($L \dashv R$). *Proof.* We verify the triangular identities governing the unit and counit of the adjunction, which we first provide. By Lemma 3.7.10, we take the natural transformation $1_{\mathbf{Rel}} \Rightarrow RL$ we take to be the identity morphism:

$$\eta_X := 1_X$$

The counit natural transformation $LR \Rightarrow 1_{\mathbf{TopRel}}$ we define to be a coarsening, the existence of which in **TopRel** is granted by Lemma 3.7.12.

$$\epsilon_{X^\tau} : X^\star \to X^\tau := \{(x, x) : x \in X\}$$

First we evaluate $L \xrightarrow{L\eta} LRL \xrightarrow{\epsilon L} L$ at an arbitrary object (set) $X \in$ **Rel**. $L(X) = X^\star = LRL(X)$, where the latter equality holds because LR is precisely the identity functor on **Rel**. For the first leg from the left, $L(\eta_X) = L(1_X) = X^\star \xrightarrow{1_X} X^\star = 1_{X^\star}$. For the second, $\epsilon_{L(X)} = \epsilon_{X^\star} = X^\star \xrightarrow{1_X} X^\star = 1_{X^\star}$. So we have that $L\eta; \epsilon L = L$ as required.

Now we evaluate $R \xrightarrow{\eta R} RLR \xrightarrow{R\epsilon} R$ at an arbitrary object (topological space) $X^\tau \in$ **TopRel**. $R(X^\tau) = X = RLR(X^\tau)$, where the latter equality again holds because $LR = 1_{\mathbf{Rel}}$. For the first leg from the left, $\eta_{R(X^\tau)} = \eta_X = 1_X$. For the second, $R(\epsilon_{X^\tau}) = R(X^\star \xrightarrow{1_X} X^\tau) = X \xrightarrow{1_X} X = 1_X$. So $\eta R; R\epsilon = R$, as required. $\square$

THE USUAL FORGETFUL FUNCTOR FROM **TopRel** TO **Loc** HAS NO LEFT ADJOINT

Just as the forgetful functor from **TopRel** to **Rel** "forgets topology while keeping the points", we might consider a forgetful functor to **Loc** that "forgets points while remembering topology". But we show that there is no such functor that forms a free-forgetful adjunction.

Reminder 3.7.14 (The category **Loc**). [] A *frame* is a poset with all joins and finite meets satisfying the infinite distributive law:

$$x \wedge (\bigvee_i y_i) = \bigvee_i (x \wedge y_i)$$

A *frame homomorphism* $\phi : A \to B$ is a function between frames that preserves finite meets and arbitrary joins, i.e.:

$$\phi(x \wedge_A y) = \phi(x) \wedge_B \phi(y) \qquad \phi(x \vee_A y) = \phi(x) \vee_B \phi(y)$$

The category **Frm** has frames as objects and frame homomorphisms as morphisms.

The category **Loc** is defined to be **Frm**$^{\text{op}}$.

Remark 3.7.15. Here are informal intuitions to ease the definition. The lattice of open sets of a given topology ordered by inclusion forms a frame – observe the analogy "arbitrary unions" : "all joins" :: "finite intersections" : "finite meets". Closure under arbitrary joins guarantees a maximal element corresponding to the open set that is the whole space. So frames are a setting to speak of topological structure alone, without referring to a set of underlying points, hence, pointless topology. Observe that in the definition of continuous functions, open sets in the *codomain* must correspond (uniquely) to open sets in the *domain* – so every continuous function induces a frame homomorphism going in the opposite direction that the function does between spaces, hence, to obtain the category **Loc** such that directions align, we reverse the arrows of **Frm**. Observe that continuous relations induce frame homomorphisms in the same way. These observations give us insight into how to construct the free and forgetful functors.

Definition 3.7.16 ($U : $ **TopRel** $\to$ **Loc**). On objects, U sends a topology X^τ to the frame of opens in τ, which we denote $\hat\tau$.

On morphisms $R : X^\tau \to Y^\sigma$, the corresponding partial frame morphism $\hat\tau \leftarrow \hat\sigma$ (notice the direction reversal for **Loc**), we define to be $\{(U_{\in\sigma}, R^\dagger(U)_{\in\tau}) \mid U \in \sigma\}$. We ascertain that this is (1) a function that is (2) a frame homomorphism. For (1), since the relational converse picks out precisely one subset given any subset as input, these pairs do define a function. For (2), we observe that the relational converse (as all relations) preserve arbitrary unions and intersections, i.e. $R^\dagger(\bigcap_i U_i) = \bigcap_i R^\dagger(U_i)$ and $R^\dagger(\bigcup_i U_i) = \bigcup_i R^\dagger(U_i)$, so we do have a frame homomorphism.

Associativity follows easily.

Proposition 3.7.17 (*U* has no left adjoint). *Proof.* Seeking contradiction, if U were a right adjoint, it would preserve limits. The terminal object in **Loc** is the two-element lattice $\bot < \top$, where the unique frame homomorphism to any $\mathcal{L}$ sends $\top$ to the top element of $\mathcal{L}$ and $\bot$ to the bottom element. In **TopRel**, the empty topology $\mathbf{0} = (\emptyset, \{\emptyset\})$ is terminal (and initial). However, $U\mathbf{0}$ is the singleton lattice, not $\bot < \top$ (which is the image under U of the singleton topology). $\qquad \square$

This is a rather frustrating result, because U does turn continuous relations into backwards frame homomorphisms on lattices of opens; see Proposition 3.4.14, and note that in the frame of opens associated with a topology, the empty set becomes the bottom element. The obstacle is the fact that the empty topology is both initial and terminal in **TopRel**. We may be tempted to try treating U as a right adjoint going to **Frm** instead, but then the monad induced by the injunction on **Loc** would trivialise: left adjoints preserve colimits, so any putative left adjoint F must send $\bot < \top$ (initial in **Frm** by duality) to the empty toplogy, and the empty topology as terminal object must be sent to the terminal singleton frame, which implies that the monad UF on **Frm** sends everything to the singleton lattice.

3.7.5   Why not Span(**Top**)?

One common generalisation of relations is to take spans of monics in the base category []. This actually produces a different category than the one we have defined. Below is an example of a span of monic continuous functions from **Top** that corresponds to a relation that doesn't live in **TopRel**. It is the span with the singleton as apex, with maps from the singleton to the closed points of a two Sierpiński spaces.
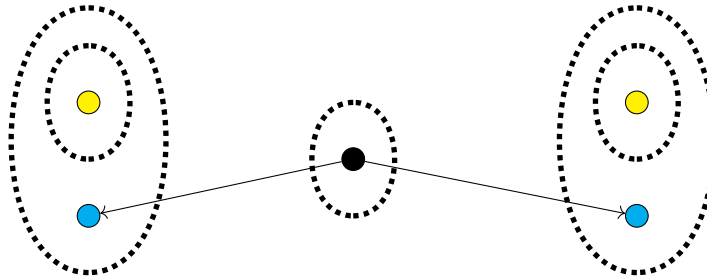
3.7.6   Why not a Kleisli construction on **Top**?

Another way to view the category **Rel** is as the Kleisli category $K_{\mathcal{P}}$ of the powerset monad on **Set**; that is, every relation $A \to B$ can be viewed as a function $A \to \mathcal{P}B$, and composition works by exploiting the monad multiplication: $A \xrightarrow{f} \mathcal{P}B \xrightarrow{\mathcal{P}g} \mathcal{P}\mathcal{P}C \xrightarrow{\mu_{\mathcal{P}C}} \mathcal{P}C$. So it is reasonable to investigate whether there is a monad T on **Top** such that K_T is equivalent to **TopRel**. We observe that the usual free-forgetful adjunction between **Set** and **Top** sends the former to a full subcategory (of continuous functions between discrete topologies) of the latter, so a reasonable coherence condition we might ask for the putative monad T to satisfy is that it is related to $\mathcal{P}$ via the free-forgetful adjunction. This amounts to asking for the following commutative diagram (in addition to the usual ones stipulating that T and $\mathcal{P}$ are monadic):

$$
\begin{array}{ccc}
\textbf{Top} & \xrightarrow{\quad T \quad} & \textbf{Top} \\
\big\uparrow\big\downarrow{\dashv} & & \big\uparrow\big\downarrow{\dashv} \\
\textbf{Set} & \xrightarrow{\quad \mathcal{P} \quad} & \textbf{Set}
\end{array}
$$

This condition would be nice to have because it witnesses $K_{\mathcal{P}}$ as precisely K_T restricted to the discrete topologies, so that T really behaves as a conservative generalisation of the notion of relations to accommodate topologies. As a consequence of this condition, we may observe that discrete topologies $X^{\star}$ must be sent to discrete topologies on their powerset $\mathcal{P}X^{\star}$. In particular, this means the singleton topology is sent to the the discrete topology on a two-element set; $T\bullet = \mathbf{2}$. This sinks us. We know from Proposition 3.4.4 that the continuous relations $X^{\tau} \to \bullet$ are precisely the open sets of τ, which correspond to continuous functions into Sierpiński space $X^{\tau} \to \mathbb{S}$, and $\mathbb{S} \neq \mathbf{2}$.

3.7.7   Where is the topology coming from?

It is category-theoretically natural to ask whether **TopRel** is "giving topology to relations" or "powering up topologies with relations", but we have explored those techniques and it doesn't seem to be that. It is possible that the failure of these regular avenues may explain why I had such difficulty finding any trace of **TopRel** in the literature. However, we do have a free-forgetful adjunction between **TopRel** and **Rel**, and if we focus on this, it *is* possible to crack the nut of where topology is coming from with enough machinery; here is one such sketch. Observe that the forgetful functor looks like it could be a kind of fibration, where the elements of the fibre over any set A in **Rel** correspond to all possible topologies on A. Moreover, these topologies may be partially ordered by coarseness-fineness to form a frame (though a considering it a preorder will suffice.) The fibre over a relation

$R : A \to B$ is all pairs of topologies τ, σ such that R is continuous between A^τ and B^σ. The crucial observation is that if R is continuous between τ and σ, then R will be continuous for any finer topology in the domain, $\tau \leq \tau'$, and any coarser topology in the codomain $\sigma' \leq \sigma$; that is, the fibre over R displays a boolean-valued profunctor between preorders. So **TopRel** can be viewed as the display category induced by a functor **Rel** $\to C$, where C is a category with preorders for objects and boolean-enriched profunctors as morphisms, and the functor encodes topological data by sending sets in **Rel** to preorders of all possible topologies, and relations to profunctors. I have deliberately left this a sketch because it doesn't seem worth it to view something so simple in such a complex way.

3.7.8   Why are continuous relations worth the trouble?

I'll have to refer you back to the introduction of this section. In short, because the opens of topological spaces crudely model how we talk about concepts, and the points of a topological space crudely model instances of concepts. Why this is so is best demonstrated by an illustrated example.

POINTS IN SPACE ARE A MATHEMATICAL FICTION. Useful, but a fiction. Suppose we have a point on a unit interval. Consider how we might tell someone else about where this point is. We could point at it with a pudgy appendage, or the tip of a pencil, or give some finite decimal approximation.
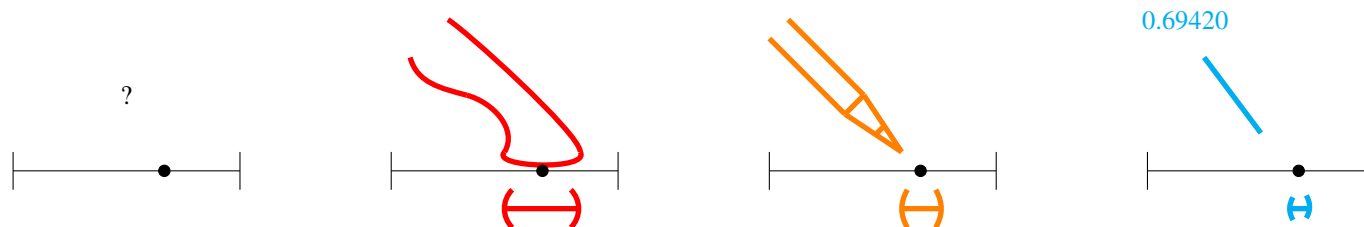
But in each case we are only speaking of a vicinity, a neighbourhood, an *open set in the borel basis of the reals* that contains the point. Identifying a true point on a real line requires an infinite intersection of open balls of decreasing radius; an infinite process of pointing again and again, which nobody has the time to do. In the same way, most language outside of mathematics is only capable of offering successively finer, finite approximations to whatever it is that occurs in the mind or in reality.

MAYBE THAT EXPLAINS THE ASYMMETRY OF WHY TESTS ARE OPEN SETS, BUT WHY ARE STATES ALLOWED TO BE ARBITRARY SUBSETS? Because states in this model represent what is conceived or perceived. Suppose we have an analog photograph whether in hand or in mind, and we want to remark on a particular shade of red in some uniform patch of the photograph. As in the case of pointing out a point on the real interval, we have successively finer approximations with a vocabulary of concepts: "red", "burgundy", "hex code #800021"... but never the point in colourspace itself. If someone takes our linguistic description of the colour and tries to reproduce it, they will be off in a manner that we can in principle detect, cognize, and correct: "make it a little darker" or "add a little blue to it". That is to say, there are, in principle, differences in mind that we cannot distinguish by boundedly finite language; we would have to continue the process of "even darker" and "add a bit less blue than last time" forever. All this is just the mathematical formulation of a very common observation: sometimes you cannot do an experience justice with words, and you eventually give up with "I guess you just had to be there". Yet the experience is there and we can perform linguistic operations on it, and the states accommodate this.

TOP IS SYMMETRIC MONOIDAL CLOSED WITH RESPECT TO PRODUCT, WHY DIDN'T YOU JUST WORK THERE FROM THE START? Because **Top** is cartesian monoidal,

which in particular means that there is only one test (the map into the terminal singleton topology), and worse, all states are tensor-separable. The latter fact means that we cannot reason natively in diagrams about correlated states, which are extremely useful representing entangled quantum states [dodo], and for reasoning about spatial relations [talkspace]. I'll briefly explain the gist of the analogy in prose because it is already presented formally in the cited works and elaborated in [bobcomp]. The Fregean notion of compositionality is roughly that to know a composite system is equivalent to knowing all of its parts, and diagrammatically this amounts to tensor-separability, which arises as a consequence of cartesian monoidality. Schrödinger suggests an alternative of compositionality via a lesson from entangled states in quantum mechanics: *perfect knowledge of the whole does not entail perfect knowledge of the parts.* Let's say we have information about a composite system if we can restrict the range of possible outcomes; this is the case for the bell-state, where we know that there is an even chance both qubits measure up or both measure down, and we can rule out mismatched measurements. However, discarding one entangled qubit from a bell-state means we only know that the remaining qubit has a 50/50 of measuring up or down, which is the minimal state of information we can have about a qubit. So we have a case where we can know things about the whole, but nothing about its parts. A more familiar example from everyday life is if I ask you to imagine `a cup on a table in a room`. There are many ways to envision or realise this scenario in your mind's eye, all drawn from a restricted set of permissable positions of the cup and the table in some room. The spatial locations of the cup and table are entangled, in that you can only consider the positions of both together. If you discard either the cup or the table from your memory, there are no restrictions about where the other object could be in the room; that is, the meaning of the utterance is not localised in any of the parts, it resides in the entangled whole.

4

Sketches of the shape of language

4.1   Lassos for generalised anaphora

A PROBLEM ARISES WHEN ANYTHING CAN BE A NOUN WIRE. By linguistic introspection, we realise we must account for *Entification* and *Processising* – the process of turning non-nouns into noun-entities and back again. If we think about English, we find that just about any word can be turned into a noun and back again (e.g. `run` by gerund to `running`, `quick` by a suffix to `quickness`, and even entire sentences `Bob drinks Duvel` can become a noun `the fact that Bob drinks Duvel`).

This consideration carries some linguistic interest as well. In the usual treatment of anaphora resolution, pronouns refer to nouns, for instance: `Bob drinks a beer.  It is cold.`, where it refers to the beer. But there are situations where pronouns can point to textual data that are not nouns. For instance: `Jono was paid minimum wage.  He didn't mind it.`, where it would like to refer to something like `the fact that Jono was paid minimum wage`. While there are extensions of discourse reference theory to accommodate event structures [], the issue at hand is that pronouns in the appropriate context seem to be able to refer to *any meaningful part of text*. For example, `The tiles were grey.  It was a particularly depressing shade.`, where it seems to refer just to the entified adjective `the greyness (of the tiles)`. Or, `Alice's cake was tastier than Bob's, but it wasn't enough so for the judges to decide unanimously.`, where *it* seems to refer the entified tastiness differential of `tastier`: `the difference in tastiness between Alice and Bob's cakes`.

Since we have so far built up a theory around noun-wires as first-class citizens, these observations present nontrivial mathematical constraints for interpretations of text circuits. Now we try to interpret these constraints in mathematical terms, staying within the graphical confines we have established in **TopRel** as much as possible. Let us denote the noun-wire type by Ξ. First we observe that any finite collection of noun wires $\bigotimes^n \Xi$ has to be *encodable* in a single noun wire Ξ, because we can always interpose with `and`. We take this to mean that there will exist morphisms such that:

placeholder

Second, for any word-gate w of grammatical type $\mathfrak{g}$, we ought to have noun-states and an evaluator process that witness entification and processising:

placeholder

Second-and-a-half, any morphism (or "meaningful part of text") $T \in \bigotimes^n \Xi$, $\bigotimes^m \Xi$ for any $n, m \in N$ – has to be encodable as a state of Ξ. This is expressed as the following graphical condition:

placeholder

Condition two-and-a-half follows if the former conditions are met, provided that all text circuits are made up of a fixed stock of grammatical-gate-types:

placeholder

If we have all the above, then we can grab any part of a circuit and turn it into a noun. We can notate this using a *lasso*.

Recall that Lassos – a graphical gadget that can encode arbitrary morphisms into a single wire – can be interpreted in a monoidal computer. Recall that monoidal computers require a universal object Ξ. Here we show how in **TopRel**, by taking $\Xi := \boxplus$ the open unit square, we have a monoidal computer in **Rel** restricted to countable sets and the relations between them. We will make use of sticky spiders. We have to show that; $\boxplus$ has a sticky-spider corresponding to every countable set; how there is a suitable notion of sticky-spider morphism to establish a correspondence with relations; what the continuous relations are on $\boxplus$ that mimic various compositions of relations.

Proposition 4.1.1 $((0, 1) \times (0, 1)$ splits through any countable set X). For any countable set X, the open unit square $\boxplus$ has a sticky spider that splits through $X^\star$.

Proof. The proof is by construction. We'll assume the sticky-spiders to be mereologies, so that cores and halos agree. Then we only have to highlight the copiable open sets. Take some circle and place axis-aligned open squares evenly along them, one for each element of X. The centres of the open squares lie on the circumference of the circle, and we may shrink each square as needed to fit all of them.

$\square$
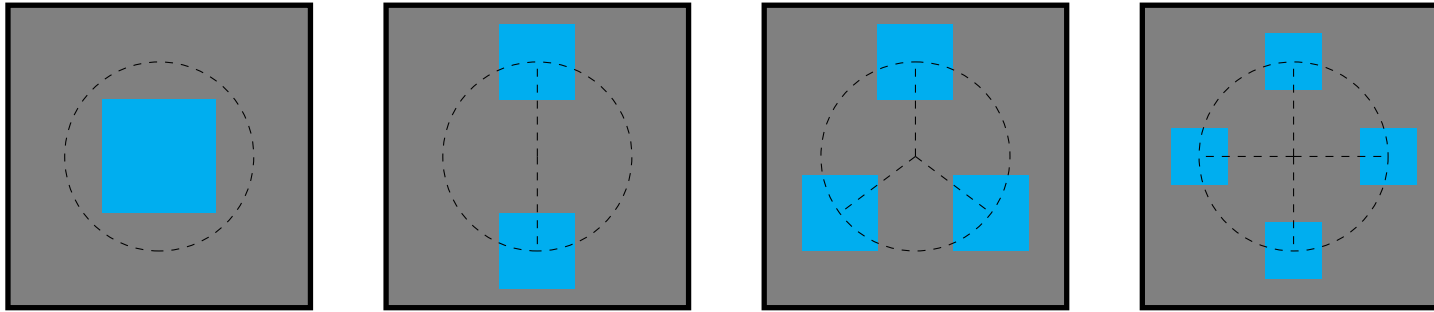
Definition 4.1.2 (Morphism of sticky spiders). A morphism between sticky spiders is any morphism that satisfies the following equation.

Proposition 4.1.3 (Morphisms of sticky spiders encode relations). For arbitrary split idempotents through $A^\star$ and $B^\star$, the morphisms between the two resulting sticky spiders are in bijection with relations $R : A \to B$.

Proof.

($\Leftarrow$) : Every morphism of sticky spiders corresponds to a relation between sets.
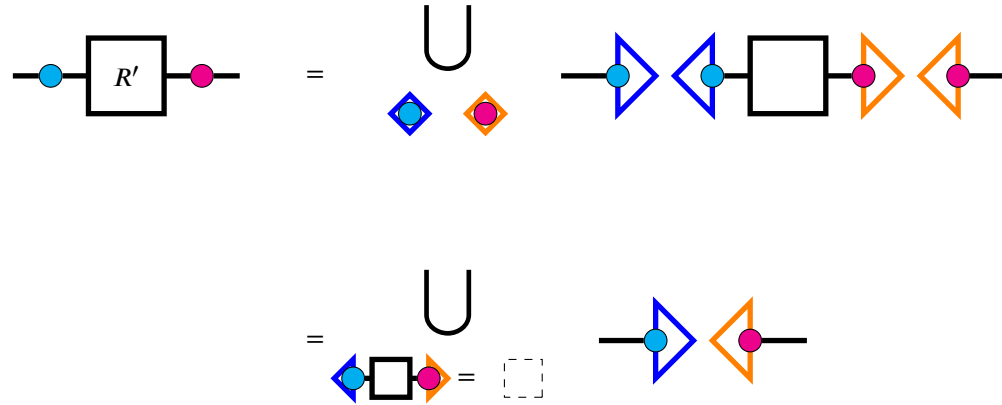
Since (co)copiables are distinct, we may uniquely reindex as:

$$= \bigcup_{(a,b) \,\in\, R \,\subseteq\, A \times B}$$
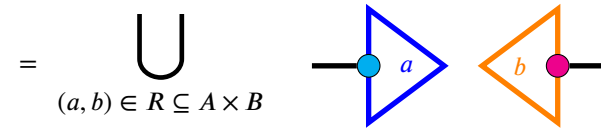
($\Rightarrow$) : By idempotence of (co)copiables, every relation $R \subseteq A \times B$ corresponds to a morphism of sticky spiders.

$$\bigcup_{(a,b) \,\in\, R}$$

$$=$$

$$\bigcup_{(a,b) \,\in\, R}$$

$\square$
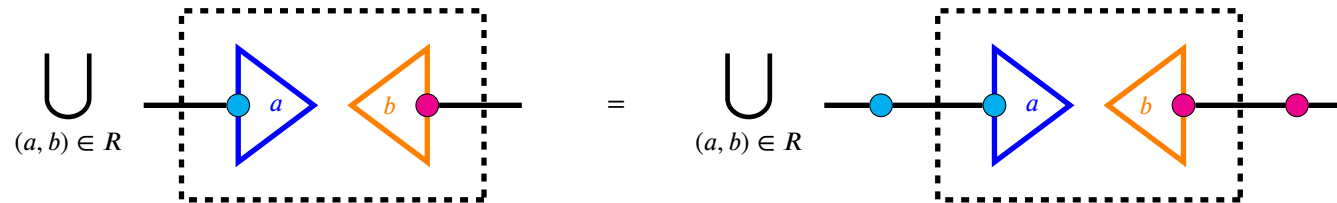
Construction 4.1.4 (Representing sets in their various guises within ▦)**.** We can represent the direct sum of two ▦-representations of sets as follows.
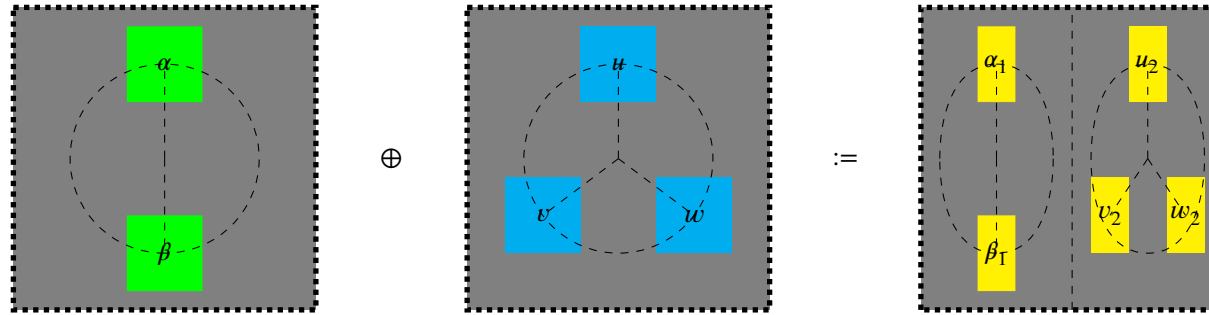
The important bit of technology is the homeomorphism that losslessly squishes the whole unit square into one half of the unit square. The decompressions are partial continuous functions, with domain restricted to the appropriate half of the unit square.

$$(x, y) \mapsto (\tfrac{x}{2}, y) \qquad (x, y) \mapsto (\tfrac{x+1}{2}, y) \qquad (x, y)|_{x<\frac{1}{2}} \mapsto (2x, y) \qquad (x, y)|_{x>\frac{1}{2}} \mapsto (2x - 1, y)$$

We express the ability of these relations to encode and decode the unit square in just either half by the following graphical equations.

Now, to put the two halves together and to take them apart, we introduce the following two relations. In tandem with the squishing and stretching we have defined, these will behave just as the projections and injections for the direct-sum biproduct in **Rel**.

The following equation tells us that we can take any two representations in ▦, put them into a single copy of ▦, and take them out again. Banach and Tarski would approve.
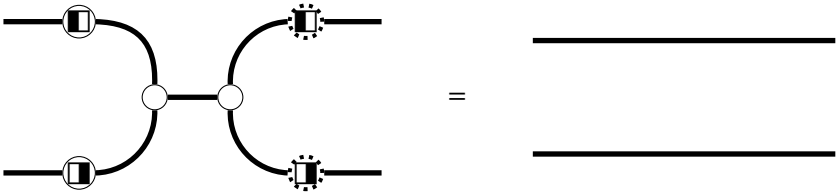
We encode the tensor product $A \otimes B$ of representations by placing copies of B in each of the open boxes of A.
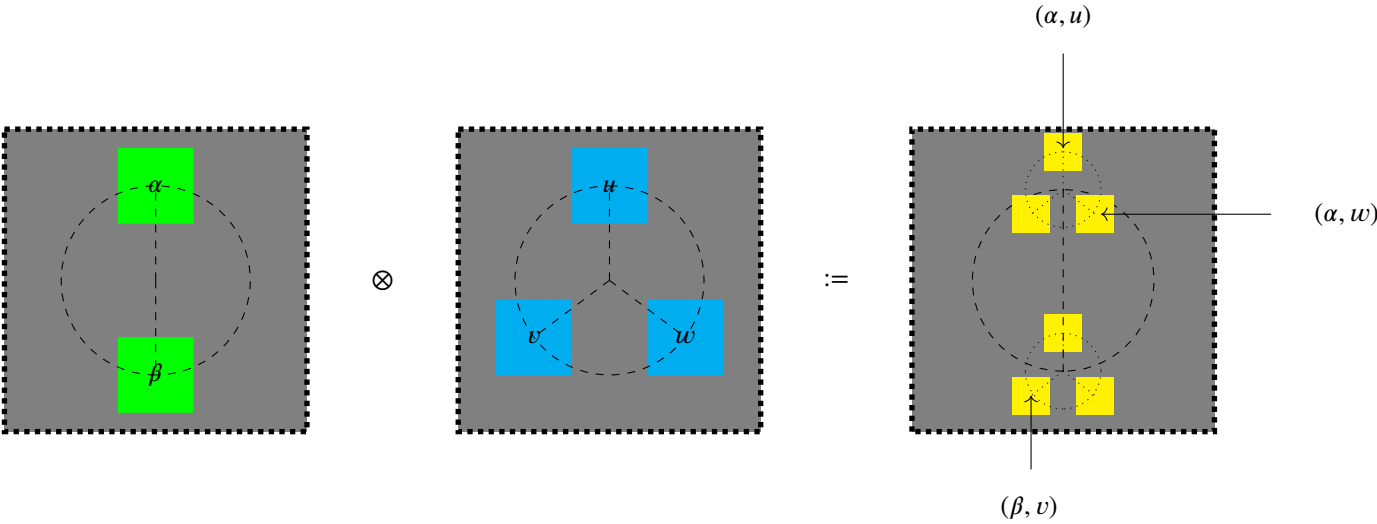
The important bit of technology here is a family of homeomorphisms of ▦ parameterised by axis-aligned open boxes. We depict the parameters outside the body of the homeomor-

phism for clarity. The squish is on the left, the stretch on the right.

$$(W, H, P_x, P_y)$$

$$(x, y) \mapsto (W\,x + P_x, H\,y + P_y)$$

$$(W, H, P_x, P_y)$$

$$(x, y)|_{\in (P_x, P_x + W) \times (P_x, P_y + H)} \mapsto (\tfrac{x - P_y}{W}, \tfrac{y - P_y}{H})$$
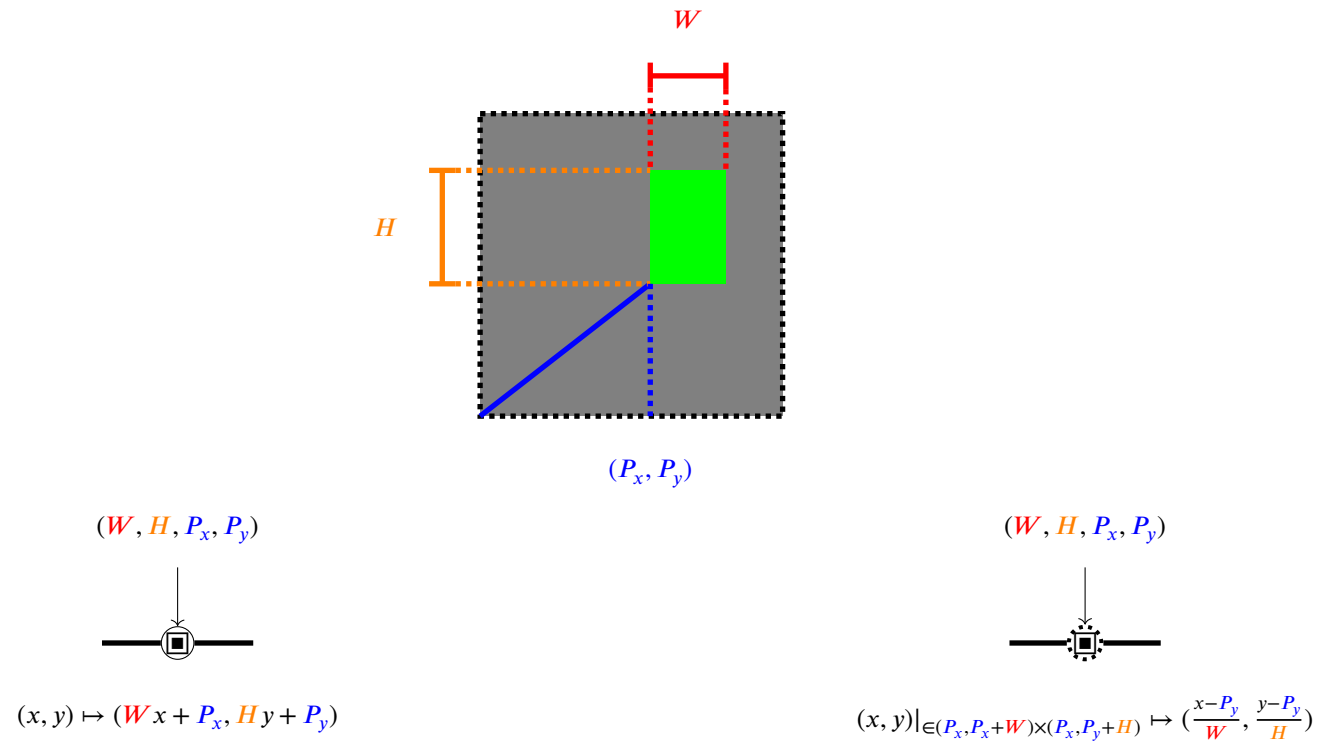
Now, for every representation of a set in ⊞ by a sticky-spider, where each element corresponds to an axis-aligned open box, we can associate each element with a squish-stretch

homeomorphism via the parameters of the open box, which we notate with a dot above the name of the element.
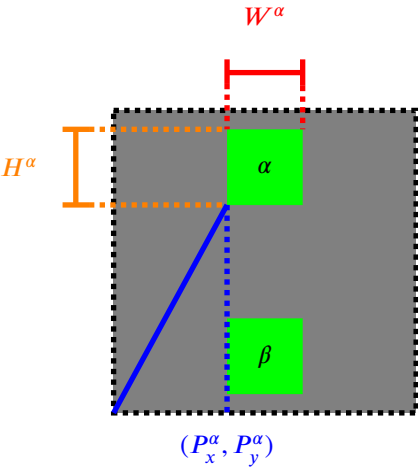
$$\dot\alpha = (W^\alpha, H^\alpha, P^\alpha_x, P^\alpha_y)$$

$$\dot\beta = (W^\beta, H^\beta, P^\beta_x, P^\beta_y)$$

Now we can define the "tensor X on the left" relation $\_ \to X \otimes \_$ and its corresponding cotensor.
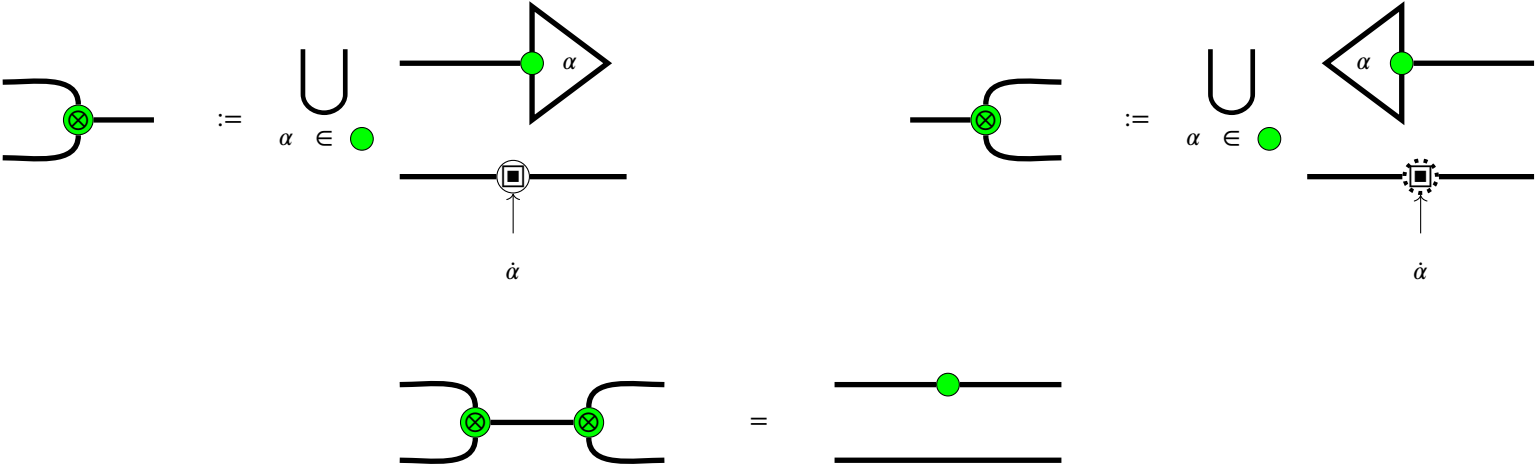
The tensor and cotensor behave as we expect from proof nets for monoidal categories.
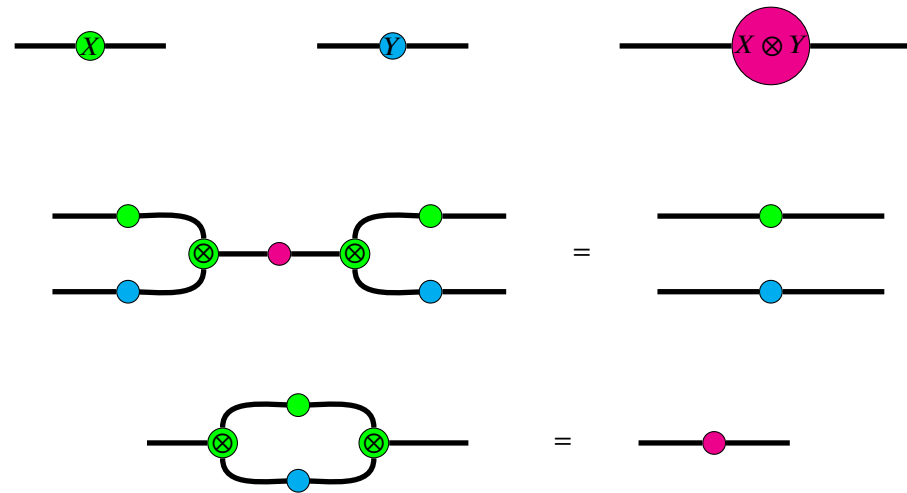
And by construction, the (co)tensors and (co)pluses interact as we expect, and they come with all the natural isomorphisms between representations we expect. For example, below we exhibit an explicit associator natural isomorphism between representations.
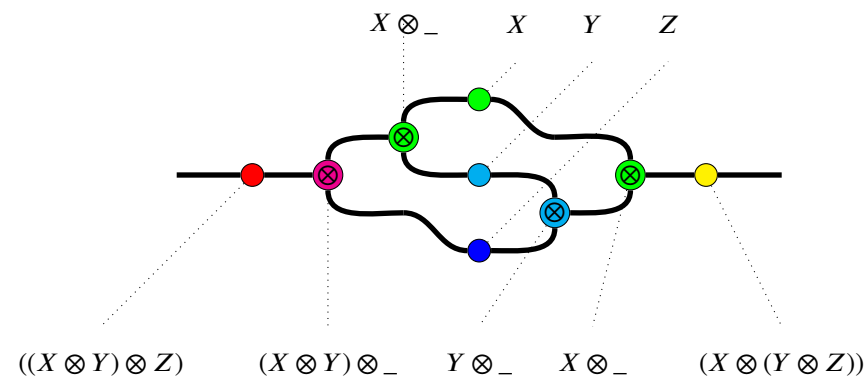
Construction 4.1.5 (Representing relations between sets and their composition within ⊞). With all the above, we can establish a special kind of process-state duality; relations as

processes are isomorphic to states of ▦, up to the representation scheme we have chosen.
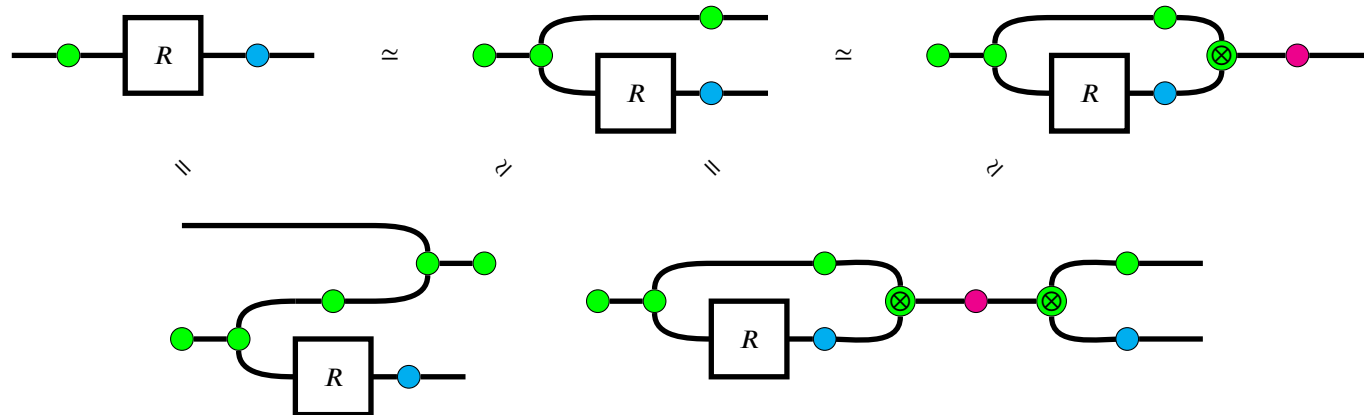
Moreover, we have continuous relations that perform sequential composition of relations.
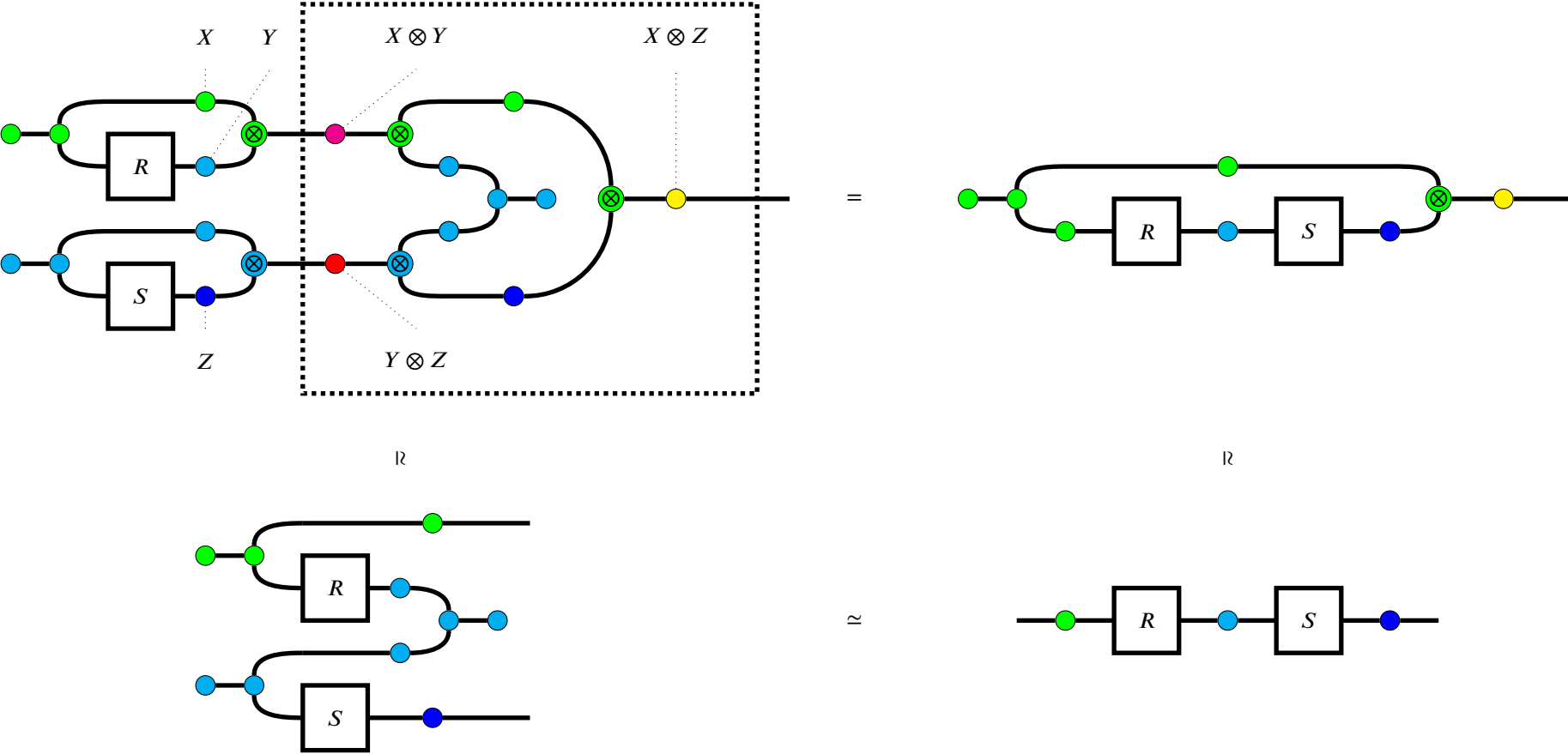
And we also know how to take the parallel composition of relations by tensors.
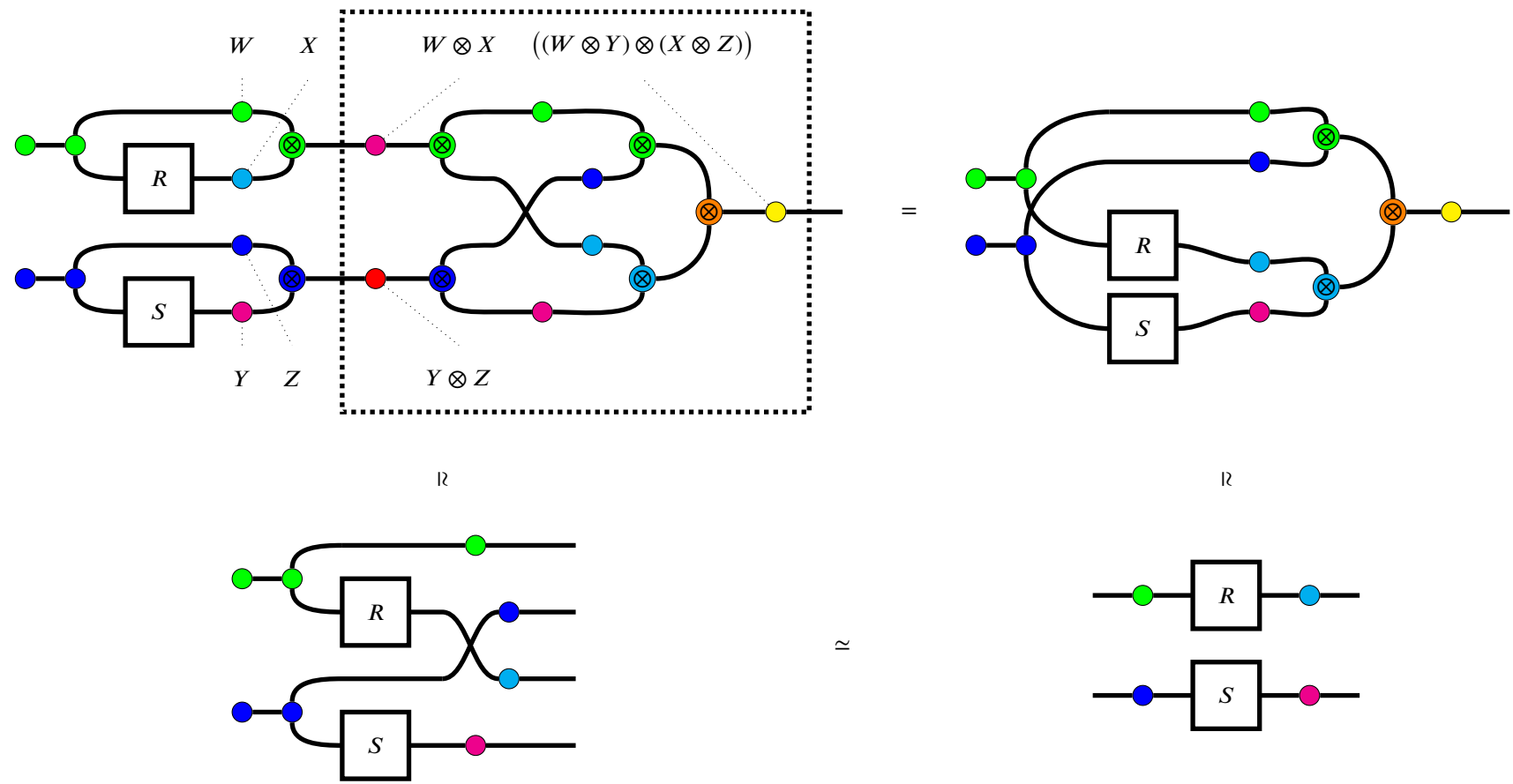
4.2   (Im)possibility results for learning text circuits from data

So far, we have been working process-theoretically, using equations between processes to specify their behaviour. It is natural to ask whether it is possible to realise each process in a process theory as a neural net, using the equations as training criteria so that the neural nets jointly model a process theory. This approach is worth pursuing to combine the benefits and ease of data-driven learning with the modularity and explainability benefits of process theories. Moreover, the onus is on us to demonstrate that text circuits can be learnt in this way, or else we would be no better off in terms of a practical theory of language for the age of big data.

In this sketch, we briefly introduce neural nets diagrammatically, along with the *Universal Approximation Theorem*, which, along with variants for different architectures, states that for any dimension m and any $\epsilon > 0$, there exists a neural net that approximates any continuous function $\mathbb{R}^m \to \mathbb{R}$ on a compact subset of the domain $\mathbb{R}^m$ within a discrepancy of ϵ. Then we introduce the notion of approximability for PROPs, and we observe that not all PROPs are approximable in terms of smooth functions of the form given by the universal approximation theorem. So we restrict our attention to PROPs for basic text circuits, which we demonstrate are suitable for certain learning tasks. We prove that basic text circuit PROPs of bounded depth and width – a notion we will define – are approximable; in other words, that text circuits work in principle alongside data-driven techniques. We close with a discussion of limitations and extensions. We give a corollary that finitely generated subcategories of **FinSet** are realisable as ensembles of deterministic neural nets, and we show how introducing probabilistic states extends the situation to **FinRel**. We formalise an observed tension between the space-resource demands of deterministic representations and unbounded compositionality by a no-go conjecture.

4.2.1   A brief summary of Neural Nets

Neural nets arise from a toy model of biological neurons. At a glance, biological neurons have many receptors and one output, and the neuron fires a signal out if its combined inputs exceed an activation threshold. As a simplification, McCulloch-Pitts neurons are a sum of n inputs passed through an activation function $\sigma : \mathbf{R}^n \to \mathbf{R}$ that is permitted to be nonlinear, but traditionally monotone increasing and sigmoidal, which bounds the range of the function $\exists a_{\mathbb{R}} b_{\mathbb{R}} \forall x_{\mathbb{R}} : a \leq \sigma(x) \leq b$, and asks that σ approaches the lower and upper bounds in the limit as x goes to $-\infty$ and ∞ respectively. Using the diagrammatic calculus for linear algebra [] equipped with a nonlinear activation function – all of which is interpretable in **TopRel**, we can immediately grasp a visual resemblance between the designs of nature and man:

placeholder

The first use of neural nets was in application to the problem of machine vision. These first, single-layer neural nets were called *perceptrons*. Mimicking the neuronal organisation of the visual cortex, it was a sensible idea to stack these layers on top of one another [] – these layers are the original reason for the word "deep" in "deep learning", but words change in meaning over time.

placeholder

The modern ubiquity of neural nets is due to several factors. First is Hinton's backpropagation algorithm [] (which may be obsolete when you are reading this by Hinton's forward-forward second salvo [].) Observe that even after one has decided on the shape of the neural net in terms of neuronal connectivity, there are still many degrees of freedom in the parameters of the activation functions, in particular their horizontal shift (bias) and vertical stretching (weights). Borrowing diagrammatic notation for parameters as orthogonal wires from [], we can depict the degrees of freedom for a single neuron like this:

placeholder

There is a massive space of parameters to set for even a moderately sized neural net, so how do we set the parameters in such a way that the neural net computes something useful? Backpropagation solves this problem by leveraging the shape of a neural net. There are many easily searchable resources that cover backpropagation for the interested reader, including category-theoretic ones []. The simple explanation goes like this. Let's just focus on the weight parameter of each neuron. By analogy each neuron is a shitty person, and their weight is how strongly they hold a binary opinion. A neural net by analogy is a shitty rigid hierachical society with voters in the back and decision makers in the front. As a simple example, Alice and Bob each make a recommendation to Claire based on what they receive as input.

placeholder

Suppose that Claire's decision is wrong. She revises her own opinion then meets with her confidantes. Alice's recommendation was faulty, so Claire blames her; as a narcissistic defense, the viciousness of the blame is proportional to how wrong Claire was. Alice revises her own opinion proportional how mean Claire is being, and then Alice goes to seek out her confidantes to perpetuate a vicious cycle of psychological violence. Bob on the other hand was right, Claire tells him this with sheepishness proportional to her error, and he starts gloating "I told you so!" with glee proportional to how much cleverer he feels than Claire. So Bob becomes slightly more entrenched in his opinion, and then he goes to seek out his confidantes to either congratulate or belittle them, again proportional to how right he was. When all of the blame and kudos has backpropagated throughout society, all the shitty people have adjusted their opinions, and their shitty society will be less prone to making the same mistake again. This process is repeated for the human equivalent of billions of years, and then you have a neural net that can recognise handwritten digits.

placeholder

All this process needs to get started is a lot of labelled pairs of data, input along with the desired output for that input. The formal terminology for the scenario above that converts data into performance is "training", which is a computationally intensive process when lots of data is involved for big neural nets. So the second factor of the ubiquity of neural nets is Moore's law and analogues, which have overseen exponential growth in computational power and digital data storage capacity. Neural nets convert data and compute power as fuel into practical applications, and we live in an era of increasingly plentiful data and compute. Hence, the bitter lesson []; clever theories are no match for stupid methods with lots of data and a big computer. But why the hell should any of this work in the first place? Surely there are limits to what neural nets can do. Now the third factor; Moore's law and the bitter lesson might be cheated, but the third factor is a law backed by mathematics.

Theorem 4.2.1 (Universal Approximation Theorem)**.**

That is, any problem that can be encoded as a continuous transformation of lists of real numbers into other lists of real numbers is potential prey for a big enough neural net. The litigious can easily spot problems in neural nets outside of this law. For example, to the best of my knowledge there is no known bound for how much data is required – as a function of desired accuracy within a desired confidence – for a neural net to learn its target accurately, so for all we know, any big neural net could suddenly fail on an easy input instance for no reason. The universal approximation theorem is a double-edged sword, and the side that cuts the holder is that for complex problems, the input data cannot span the whole problem domain, so there will be many neural nets that agree perfectly on the training data but will perform differently out-of-distribution. Now we will try to blunt the painful edge by using the universal approximation theorem to our advantage.

4.2.2   Approximating Text Circuits with deterministic neural nets

There is a lot to be gained from a process-theoretic view of interacting ensembles of neural nets. For a simple example, consider that an autoencoder is precisely a pair of neural nets trained cooperatively encode a large input space into a small latent space and decode the original input from the latent space. Diagrammatically, this amounts to asking for the equations of a split idempotent to be treated as training conditions for a pair of processes.

autoencoder

If that's what we can do with a pair of equations, what can we do with an arbitrary PROP? We first need to decide what qualifies as a valid interpretation the generators of a PROP in terms of neural nets. Not just any functor will do, because we want to rule out trivial solutions that map all processes to constant functions. We also need to put in some work to interpret what equality of processes should mean in the setting of neural nets.

Definition 4.2.2 (Approximating a (coloured) PROP). An $(\epsilon^=, \epsilon^{\neq})$-approximation of a finitely presented coloured PROP $\mathfrak{P}$ is a strict symmetric monoidal functor $\mathcal{T}$ that interprets $\mathfrak{P}$ in the (cartesian) symmetric monoidal subcategory of **Top** generated by Euclidean spaces with the usual metric as wires equipped with cartesian copy and delete, along with neural nets as processes. As a nontriviality condition, $\mathcal{T}$ must send each wire colour in $\mathfrak{P}$ to a Euclidean space of finite positive dimension. Equality relations presented in $\mathfrak{P}$ are interpreted as $\epsilon^=$-closeness by $\mathcal{T}$, i.e. if $\mathfrak{P}$ stipulates that $f = g$ for $f, g : A \to B$, then we have the following inequality in the metric of $\mathcal{T} B$:

$$\forall \mathbf{x}_{\in \mathcal{T} A} : d_{\mathcal{T} B}\big(\mathcal{T} f(\mathbf{x}), \mathcal{T} g(\mathbf{x})\big) \leq \epsilon^=$$

Any PROP that equates generators directly is redundant, and we can without loss of generality restrict consideration to PROPs where each generator is implicitly assumed to be distinct. We interpret inequality as $\epsilon^{\neq}$ farness, i.e., for all pairs of generators f, g of the same type $A \to B$, we ask that $\mathcal{T}$ satisfies:

$$\exists \mathbf{y}_{\in \mathcal{T} A} : d_{\mathcal{T} B}\big(\mathcal{T} f(\mathbf{y}), \mathcal{T} g(\mathbf{y})\big) \geq \epsilon^{\neq}$$

Since the category is cartesian monoidal, states are points in euclidean space, and the above definition specialise to treating points as "equal" if they are $\epsilon^=$-close and "inequal" if they are $\epsilon^{\neq}$-far. We choose to treat the determination of equality and inequality as separate semidecidable procedures because "equality" as we have defined it is not necessarily transitive, but we can recover a form of bounded transitivity by making $\epsilon^=$ very small compared to $\epsilon^{\neq}$, so that equality is testable within a tolerance of $\epsilon^{\neq}$, granting $\frac{\epsilon^{\neq}}{\epsilon^=}$-fold transitivity. We can always recover decidable "equality" at the expense of transitivity by setting $\epsilon^= = \epsilon^{\neq}$. With that out of the way, we observe that since the target category of deterministic neural nets is cartesian monoidal, not all PROPs are approximable.

Example 4.2.3 (Not all PROPs are approximable). We take the snake equations as an example. The PROP generating the snake equation is as follows:

snakeprop

Since we are dealing with a cartesian monoidal category, the cup can only be interpreted as a pair of points, and the cap can only be a pair of deletes []. The only Euclidean space in which the identity is equal to a constant map is the singleton zero-dimensional space.

trivialityproof

So nondeterminism is a necessary but possibly insufficient condition for the realisation of general PROPs. Not all is lost; if we restrict our consideration to well-behaved PROPs, such as those of simple text circuits, then we can get somewhere eventually.

Definition 4.2.4 (Basic Text Circuit PROP). A *basic text circuit PROP* has two colours of wires, N for "noun" and A for "answer". The generators fall under four main families. *Nouns* have type $1 \to N$. *Gates* have type $\bigotimes^k N \to \bigotimes^k N$ for some positive k. *Queries* have type $\bigotimes^k N \to A$ for some positive k. *Answers* have type $1 \to A$.

$$states, gates, queries, answers$$

The relations of a text circuit fall under three families. *Axioms* are equations between pairs of nonempty composites of gates; the only kind we disallow is an equality between two generators.

$$axiom$$

Instances are equations between a composite of nouns and gates and a single query on the left – a *datum* – and an answer on the right – a *label*.

$$instance$$

In addition, we ask for a special generator with a non-finite family of rules to enforce a coherence condition; we decide that distinct nouns should maintain their identity no matter what relations they participate in. The generator is *Name*, of type $N \to N$, and its relations are such that applying *Name* to any noun-wire that traces back to a noun will return that noun.

$$name$$

Example 4.2.5 (How basic text circuits PROPs may be used in practice).

A limitation that arises from the interaction of the definition of approximability and the universal approximation theorem is that any compact subset of euclidean space can be covered by finitely many ϵ-balls. This means that the universal approximation theorem is unable to provide us guarantees if we want potentially unboundedly large text circuits, but we might reasonably expect compositional behaviour up to some notion of text circuits with bounded width and depth, which we state as follows.

Definition 4.2.6 (Bounded width and depth). We say that a basic text circuit PROP $\mathfrak{T}$ is *terminating* if all of its axiom relations can be equipped with directions so that applying directed equality rewrites to any diagram (necessarily finite) yields a finite set of equal diagrams. As an easy example, a basic text circuit PROP with a single idempotent gate is terminating when we equip the idempotence relation with a direction that reduces the number of gates; we don't want to deal with cases where equalities explode to infinity.

$$idempotentreduce$$

Observe that if $\mathfrak{T}$ is terminating, then applying axioms to the datum of any instance relation will also yield a finite set of equal diagrams, and each instance diagram may be rewritten by isotopies so that parallel gates are displaced so that each gate occupies a distinct level, sandwiched by a layer of nouns and query, which we also count as layers. We say that $\mathfrak{T}$ has *bounded depth $d \in \mathbb{N}$* if the maximum depth in layers obtained in this way from any datum in $\mathfrak{T}$ is bounded above by d. The case of width is simpler, because axioms cannot change the number of wires in a datum, which is the same as the number of nouns; we say that $\mathfrak{T}$ has *bounded width $w \in \mathbb{N}$* if the maximum number of nouns that occur in any datum is bounded above by w.

Even this is not enough. Another issue arises from the interaction of approximability with the nature of cartesian monoidal categories.

Theorem 4.2.7 (Fox's Theorem).

A consequence of Fox's theorem is that in any cartesian monoidal category, we can equip every wire with canonical cocommutative comonoids (copy and delete), and every morphism in a cartesian monoidal category is a cohomomorphism with respect to copy and delete; recall from Section 1.2.4 that this is the diagrammatic definition of deterministic maps. This consequence means that for some text circuit PROPs, approximable-equalities may hold in *any* of their interpretations as deterministic neural nets that are not equalities licensed by the original PROP.

Example 4.2.8.

The deeper essence of this issue is that in a cartesian monoidal category, every wire can at most carry data about its diagrammatic causal past, since equalities of the following sort always hold.

placeholder

This conflicts with the nature of updating representations with text, where later information may force revisions of earlier representations. For a crude example, consider this transcription of a meme about a morning routine:

```
Wake up.  Take a shit.  Eat.  Get out of bed.  Have breakfast.
```

Now I will kill the joke by overthinking it. What happens in our mind's eye if we are constructing a little vignette of events? In this example, the first three clauses take a pedestrian interpretation – `Taking a shit` normally happens in toilets, and the inferred object of `Eat` is breakfast. Clauses four and five require belief revisions. The internal representation of the sequence of events up to those points must be retroactively changed in a manner that is not representable as gates updating entities locally:

cartoon

So in the deterministic setting, after every local update, we require a *global* coordination gate that gives all representations access to each other's data and a chance for them to revise themselves. To summarise our restrictions so far, we are only dealing with text circuit PROPs, of bounded depth to remain within the guarantees of the universal approximation theorem, and of fixed width as a diagrammatic consequence of having a single global coordinator gate.

Theorem 4.2.9 (Basic Text Circuit PROPs of bounded depth and fixed width are approximable using a global coordinator)**.** *Proof.* We sketch a construction. The essential idea is to have each noun represent all of the gates it has passed through. The representation is (the monoid analogue of) a Cayley graph of a free group in $\mathbb{R}^{|\mathbf{G}|}$, where if $w \in \mathbb{N}$ is the fixed width, $\mathbf{G}$ is the set of gates where every wire but one is assigned: $\{g(i, j, \cdots, -, \cdots, k) \mid g \in \text{Gates}, \ 1 \leq i, j, \cdots k \leq w\}$. In the illustration below, we can for instance record that a noun wire has passed through a unary gate h as the only argument, or that it has passed through three gates, first a binary f gate as the first argument, sharing the gate with the third wire, then a binary g gate as the second argument, sharing the gate with the first wire, and then a unary h gate. Each distinct generator has its own dimension in Euclidean space.
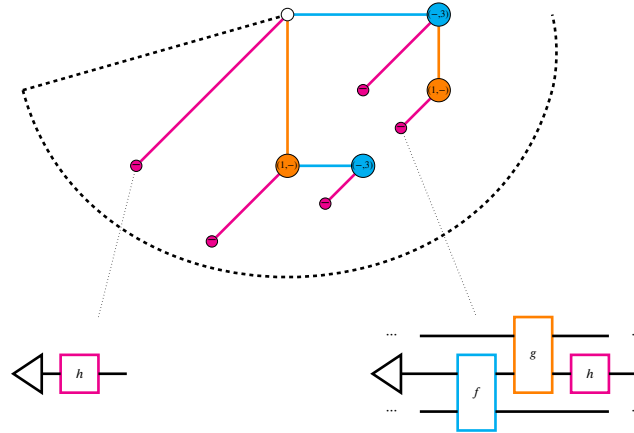
By asking for each node to be an ϵ-ball, we can test for equality by a spike function.

$$\mathbf{S}(x, w, h) \quad :=$$
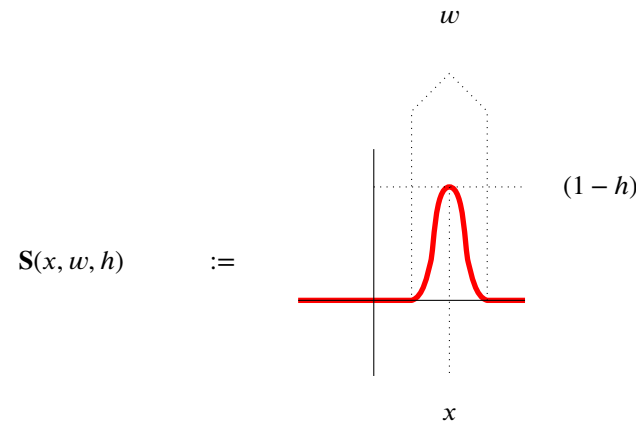
   We don't want equal steps in each direction in the graph, or else all pairs of steps would commute. To reflect the order in which steps occur, we want subsequent steps to be smaller than earlier ones. So for each colour of node in the graph, we add the value of a move function in the desired dimension, parameterised by the initial position of the noun, so that each subsequent step is half the distance of the previous one. To make life easier, we have each move function use the taxicab metric. By bounded depth, there is some $d \in \mathbb{N}$ after which we don't need to carry representations, so we modify the ends of the move function to hit zero early. Setting $R \gg 2^d \epsilon$, each move function acts to translate all graph nodes within an R-ball to their destination graph nodes after uniformly applying some generator. We ask that each wire is initialised at least $2R$-far apart, so each wire has their own R-ball neighbourhood in which to encode data as a graph.
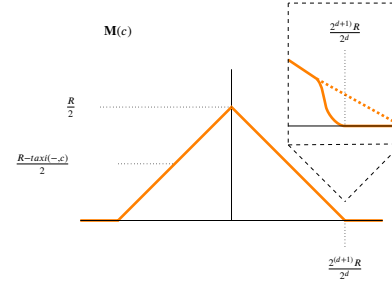
Now we can jointly specify representations of nouns and gates. For example, suppose f is a binary gate, and that $w = 3$. Then there are six shift functions

$$\mathbf{M}_f(-, 1), \mathbf{M}_f(-, 2), \mathbf{M}_f(-, 3), \mathbf{M}_f(1, -), \mathbf{M}_f(2, -), \mathbf{M}_f(3, -)$$

Set an ansatz dimension $\mathcal{A}$ aside, which distinguishes noun wires by indicator-spike functions $\mathbf{1}_1, \mathbf{1}_2, \mathbf{1}_3$ – a continuous analog of one-hot encoding – then let f be:

$$f(x, y) := \begin{cases} x \mapsto x + \sum\limits_{i^x, i^y = 1}^{w} \left(\mathbf{1}_{i^x}(x) \mathbf{1}_{i^y}(y) \mathbf{M}_f(-, i^y) \right) \\ y \mapsto y + \sum\limits_{i^x, i^y = 1}^{w} \left(\mathbf{1}_{i^x}(x) \mathbf{1}_{i^y}(y) \mathbf{M}_f(i^x, -) \right) \end{cases}$$

In prose, the coefficients obtained by the indicator functions of each summand behave as a case guard, identifying the wire-numbers of x, y by sending all other cases to zero. This generalises to representations of gates with higher arity. At this point, we have obtained representations for a free theory of just the gates of bounded-depth and fixed-width text circuits – we have not yet incorporated the potential equalities of the theory. First we deal with the axiom-type equalities. Here is where we make use of a global coordinator gate; where width is w and each noun is represented in some space $\mathbb{R}^J$, the global coordinator is typed $\prod\limits^{w} \mathbb{R}^J \to \prod\limits^{w} \mathbb{R}^J$. The global coordinator looks at the local data recorded by each wire, and when it spots an equality in the theory, it pinches free graph nodes together. The pinch function is a modification of the spike and shift functions, moving only points within a tolerance β into a target ϵ-ball.
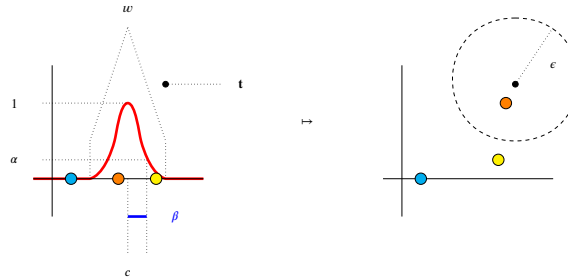
The global coordinator waits to see if the product state of all the wires satisfies an equality, and when it does, it pinches the agreeing free representations on each wire to a new node along a new dimension – we ask for one dimension for each equivalence class of expressions quotiented by equality in the theory, and here the bounded-depth and fixed-width conditions mean that we only have to add finitely more dimensions in this way.
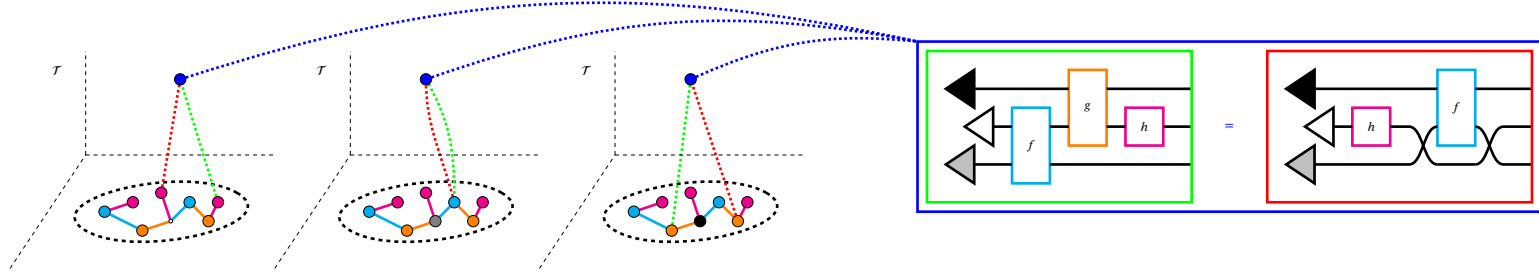
The target location of each pinch can be chosen to be compatible with the shift functions that generate the graphs – i.e. each equality-class is treated as if it were its own generator with its own shift function. The global coordinator is the sum of all such pinches, and the way to use it is to sandwich it between each gate.

placeholder

Finally, the query morphisms in the instance-type rules of the text circuit PROP are continuous functions that assign each ϵ-ball node a value in a Euclidean answer-space, which can mimic a finite discrete number of cases by spike-indicator functions. All of this is, by construction, an (ϵ, R)-approximation of a text circuit PROP. $\square$

4.2.3 Text circuits of unbounded width in noncartesian settings

4.2.4 Text circuits with unbounded depth and width

Definition 4.2.10 (Faithful models and expressive completeness)**.** Given a coloured prop $\mathfrak{P}$, a symmetric monoidal category $\mathcal{M}$, and a symmetric monoidal functor $T : \mathfrak{P} \to \mathcal{M}$, we say that T is a *faithful model of* $\mathfrak{P}$ *in* $\mathcal{M}$ if, for all diagrams $f, g \in \mathfrak{P}$, $f =_{\mathfrak{P}} g \iff T(f) =_{\mathcal{M}} T(g)$. Given a collection of props $\{\mathfrak{P}\}$, we say that $\mathcal{M}$ is *expressively complete for* $\{\mathfrak{P}_i\}$ when each $\mathfrak{P}_i$ has a faithful model in $\mathcal{M}$.

Theorem 4.2.11 (**Rel**$^{\times}$ is expressively complete for all finitely presented coloured PROPs)**.** *Proof.* Our strategy has two main ideas. First is to take the Lindenbaum-Tarski algebra of diagrams for an arbitrary $\mathfrak{P}$, quotiented by the equivalence relation $=_{\mathfrak{P}}$; this will give faithfulness. Second is to treat the sought functor T as a category of elements construction, adapted to the symmetric monoidal setting.

Let $\mathfrak{P}^{\star}$ denote the finitely presented coloured PROP $\mathfrak{P}$ augmented with new generators that do not obey any relations: a state $\circ\!\!-$ and effect $-\!\!\circ$ for each colour C in $\mathfrak{P}$. Let $\mathfrak{P}^{\star}_{\mathbf{LT}}$ denote the set of diagrams of $\mathfrak{P}^{\star}$ quotiented by the equivalence relation obtained by equality in $\mathfrak{P}$, $=_{\mathfrak{P}}$. For each colour $C \in \mathfrak{P}$, let $T(C)$ be the subset of $\mathfrak{P}^{\star}_{\mathbf{LT}}$ that selects all states on C. Let $T(0)$ be the singleton. For each nonempty list of colours $[C_i]$, let $T([C_i])$ be the subset of $\mathfrak{P}^{\star}_{\mathbf{LT}}$ that selects all states on $C_1 \otimes_{\mathfrak{P}} C_2 \otimes_{\mathfrak{P}} \cdots C_i$. For $f : C \to D$ in $\mathfrak{P}$, let $T(f) := \{(\ulcorner\langle c|\urcorner, \ulcorner\langle c| ; f\urcorner) \mid c_{\in T(C)}\}$, where corner notation denotes an equivalence class of states under $=_{\mathfrak{P}}$ – note that applying f after any state $\ulcorner\langle c|\urcorner \in T(C)$ yields a state

$\ulcorner \langle c| \, ; f \urcorner \in T(D)$), and that these states include actual states native to $\mathfrak{P}$ and formal states from $\mathfrak{P}^\star$ where diagrams from $\mathfrak{P}$ with an output wire typed C have their other wires "capped off" by $\multimap$ and $\circ\!\!-$.

T is a functor; $T(1_C) = 1_{T(C)}$ since $\{(\ulcorner\langle c|\urcorner, \ulcorner\langle c|\, ; 1_C\urcorner) \mid \ulcorner\langle c|\urcorner_{\in T(C)}\}$ is the identity relation on $T(C)$; $T(f) \, ;_{\mathbf{Rel}} \, T(g) = T(f \, ;_{\mathfrak{P}} \, g)$ since the relational composite is

$$T(f) \, ;_{\mathbf{Rel}} \, T(g) := \{(\ulcorner\langle c|\urcorner, \ulcorner\langle e|\urcorner) \mid c_{\in T(C)} \, , \, \exists \ulcorner\langle d|\urcorner_{\in T(D)} : \ulcorner\langle c|\, ; f\urcorner = \ulcorner\langle d|\urcorner \& \ulcorner\langle d|\, ; g\urcorner = \ulcorner\langle e|\urcorner\}$$

We observe that $\ulcorner\langle d|\urcorner_{\in T(D)} : (\ulcorner\langle c|\, ; f\urcorner = \ulcorner\langle d|\urcorner \& \ulcorner\langle d|\, ; g\urcorner = \ulcorner\langle e|\urcorner)$ implies that $\ulcorner\langle c|\, ; f; g\urcorner = \ulcorner\langle e|\urcorner$, so we have $T(f \, ;_{\mathfrak{P}} \, g \subseteq T(f) \, ;_{\mathbf{Rel}} \, T(g)$. For the other inclusion, we observe that $\ulcorner(\langle c|\, ; f); g\urcorner$ yields the state $\ulcorner\langle c|\, ; f\urcorner \in T(D)$ in the bracketed expression, thus satisfying the existential quantifier.

The unitors, associators, braidings, and coherences are tedious to write but conceptually trivial, so I will skip them. The tricky part of showing that T is monoidal is providing isomorphisms $T(C) \times T(D) \simeq T(C \otimes_{\mathfrak{P}} D)$. We present them first and comment later.

$$T(C) \times T(D) \to T(C \otimes_{\mathfrak{P}} D) := \{(\begin{pmatrix} \ulcorner\langle c|\urcorner \\ \ulcorner\langle d|\urcorner \end{pmatrix}, \begin{cases} \ulcorner\langle x|\urcorner & \text{if } \exists \langle x|_{\in T(C \otimes_{\mathfrak{P}} D)} : \ulcorner\langle c|\, ; \multimap_C\urcorner = \ulcorner\langle x|\, ; (\multimap_C \otimes_{\mathfrak{P}} \multimap_D)\urcorner = \ulcorner\langle d|\, ; \multimap_D\urcorner \\ \ulcorner\langle c| \otimes_{\mathfrak{P}} \langle d|\urcorner & \text{otherwise} \end{cases}) \mid \ulcorner\langle c|\urcorner_{\in T(C)}, \ulcorner\langle d|\urcorner_{\in T(D)}\}$$

$$T(C \otimes_{\mathfrak{P}} D) \to T(C) \times T(D) := \{\ulcorner\langle x|\urcorner, \begin{cases} \begin{pmatrix} \ulcorner\langle c|\urcorner \\ \ulcorner\langle d|\urcorner \end{pmatrix} & \text{if } \exists \ulcorner\langle c|\urcorner_{\in T(C)} \ulcorner\langle d|\urcorner_{\in T(D)} : \ulcorner\langle x|\urcorner = \ulcorner\langle c| \otimes_{\mathfrak{P}} \langle d|\urcorner \\ \begin{pmatrix} \ulcorner\langle x|\, ; (1_C \otimes_{\mathfrak{P}} \multimap_D)\urcorner \\ \ulcorner\langle x|\, ; (\multimap_C \otimes_{\mathfrak{P}} 1_D)\urcorner \end{pmatrix} & \text{otherwise} \end{cases}) \mid \ulcorner\langle x|\urcorner_{\in T(C \otimes_{\mathfrak{P}} D)}\}$$

We walk through the construction case-by-case. For the first case top-to-bottom, if two states $\langle c|, \langle d|$ are obtainable by formally deleting the C and D outputs (using $\multimap$) of some state $\langle x|$ on $C \otimes_{\mathfrak{P}} D$, then we send the pair $(\langle c|, \langle d|)$ to $\langle x|$, quotienting by $=_{\mathfrak{P}}$. Note that in the first case, even if $\langle x|$ is tensor separable as $\langle c| \otimes_{\mathfrak{P}} \langle d|$, formal deletion creates new formal scalars which must also agree by the case guard. The total effect of the first case is to identify when $(\langle c|, \langle d|)$ arise as partial diagrams (where ends are truncated with $\multimap$) of some state $\langle x|$. The second case is where $(\langle c|, \langle d|)$ are not partial diagrams in this way, in which case we send the pair to their tensor product. The third case is the inverse of the second, and the fourth case is the inverse of the first. The effect of taking equivalence classes makes the first relation as a whole an injective function, and likewise for the second relation.

All that remains is to show that T is a faithful model, i.e., for all lists of colours $[C], [D]$ and for all $f, g : [C] \to [D]$, $f =_{\mathfrak{P}} g \iff T(f) =_{\mathbf{Rel}} T(g)$. For the forward direction $\Rightarrow$, if $f =_{\mathfrak{P}} g$, then for any state $\langle x| : 0 \to [C]$, we have that $\langle x|\, ; f = \langle x|\, ; g$. So, setting $T(\langle x|) = \ulcorner\langle x|\urcorner \in T([C])$, we have $\ulcorner\langle x|\, ; f = \langle x|\, ; g\urcorner \in T([D])$, thus $T(f) =_{\mathbf{Rel}} T(g)$. For the converse direction $\Leftarrow$, if $f \neq_{\mathfrak{P}} g$, then by the Lindenbaum-Tarski construction and the relational-freeness of the generator $\circ\!\!-$, $\ulcorner\circ\!\!-; f\urcorner \neq \ulcorner\circ\!\!-; g\urcorner$, so $T(f)$ as a relation contains the pair $(\ulcorner\circ\!\!-_{[C]}\urcorner, \ulcorner\circ\!\!-_{[C]}; f\urcorner)$ that is not present in $T(g)$, and symmetrically for $(\ulcorner\circ\!\!-_{[C]}\urcorner, \ulcorner\circ\!\!-_{[C]}; g\urcorner)$, thus $T(f) \neq T(g)$. □

4.2.5   Discussion

Example 4.2.12.

4.3   Modelling metaphor

I will take a *metaphor* to be text where systematic language for one kind of concept is used to structure meanings for another kind of concept where literal meanings cannot apply. This may subsume some cases of what would otherwise be called *similes* or *analogies*.

THE IDEA: First, we observe that we can model certain kinds of analogies between conceptual spaces by considering structure-preserving maps between them. For example, Planck's law gives a partial continuous function from part of the positive reals measuring Kelvin to wavelengths of light, and the restriction of this mapping to the visible spectrum gives the so-called "colour temperature" framework used by colourists.

Second, we observe that we can use simple natural language to describe conceptual spaces, instead of geometric or topological models. Back to the example of colour temperature, instead of precise values in Kelvin, we may instead speak of landmark regions that represent both temperature and colour such as "candle", "incandescent", "daylight", which obey both temperature-relations (e.g. candle is a lower temperature than incandescent) and colour-relations (daylight is bluer than incandescent).

Third, we observe that we can also use simple natural language to describe more complex conceptual schemes with interacting agents, roles, objects, and abilities. Organising this linguistic data in the concrete structure of a text circuit allows us to formally specify what it means for one conceptual scheme to structure another by describing structure-preserving maps between the text circuits. This will allow us to reason about and calculate with metaphors such as "Time is Money".

THE MOTIVATION:

4.3.1   Orders, Temperature, Colour, Mood

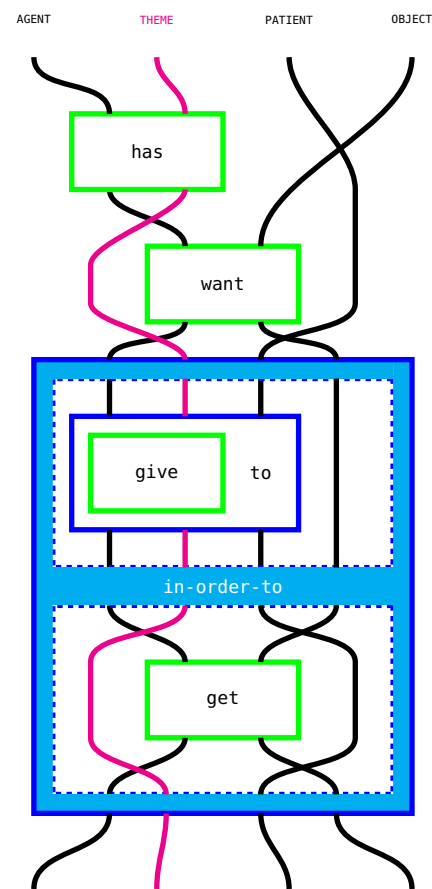### 4.3.2   Complex conceptual structure

TIME IS MONEY
```
"Do you have time to look at this?"
"This is definitely worth the time!"
"What a waste of time."
```
I will work through an example of partially using the concept of Money to structure that of Time. Part of the concept of Money is that it can be *exchanged* for something. The concept of exchange can be glossed approximately as the following text, with variable noun-entries capitalised.

```
AGENT has THEME.
AGENT wants OBJECT.
AGENT gives THEME to PATIENT in-order-to get OBJECT.
```


The

4.3.3