VINCENT WANG-MAŚCIANICA

STRING DIAGRAMS FOR TEXT

Contents

(Acknowledgements will go in a margin note here.)

Definition 0.1.1 (Lexicon)**.** We define a limited lexicon $\mathcal{L}$ to be a tuple of disjoint finite sets $(\mathbf{N}, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_S, \mathbf{A}_N, \mathbf{A}_V, \mathbf{C})$

Where:

- $\mathbf{N}$ is a set of *proper nouns*
- $\mathbf{V}_1$ is a set of *intransitive verbs*
- $\mathbf{V}_2$ is a set of *transitive verbs*
- $\mathbf{V}_S$ is a set of *sentential-complement verbs*
- $\mathbf{A}_N$ is a set of *adjectives*
- $\mathbf{A}_V$ is a set of *adverbs*
- $\mathbf{C}$ is a set of *conjunctions*

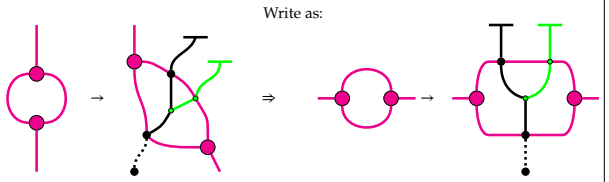


Figure 1: **How to read the diagrams in this section:** we will be making heavy use of pink and purple bubbles as frames to construct circuits. We will depict the bubbles horizontally, as we are permitted to by compact closure, or by reading diagrams with slightly skewed axes.

0.1 A generative grammar for text circuits

0.1.1 A circuit-growing grammar

There are many different ways to write a weak n-categorical signature that generates circuits. Mostly as an illustration of expressivity, I will provide a signature where the terms "surface" and "deep" structure are taken literally as metaphors; the generative grammar will grow a line of words in syntactic order, and like mushrooms on soil, the circuits will behave as the mycelium underneath the words. It won't be the most efficient way to do it in terms of the number of rules to consider, but it will look nice and we'll be able to reason about it easily.

Simplifications and limitations: For now we only consider word types as in Definition 0.1.1, though we will see how to engineer extensions later. We only deal with propositional statements, without determiners, in only one tense, with no morphological agreement between nouns and their verbs and referring pronouns, and we assume that adverbs, adjectives stack indefinitely and without further order requirements: e.g. `Alice happily secretly finds red big toy shiny car that he gives to Bob` is a sentence we consider grammatical enough. For now, we consider only the case where adjectives and adverbs appear before their respective noun or verb. Note that all of these limitations apart from the limited lexicon can principle be overcome by the techniques we developed in Section **??** for restricted tree-adjoining and links. As a historical remark, generative-transformational grammars fell out of favour linguistically due to the problem of overgeneration: the generation of nonsense or unacceptable sentences in actual language use. We're undergenerating and overgenerating at the same time, but we're also not concerned with empirical capture: we only require a concrete mathematical basis to build interesting things on top of. On a related note, there's zero chance that this particular circuit-growing grammar even comes close to how language is actually produced by humans, and I have no idea whether a generalised graph-rewriting approach is cognitively realistic.

Mathematical assumptions: We work in a dimension where wires behave symmetric monoidally by homotopy, and further assume strong compact closure rewrite rules for all wire-types. Our strategy will be to generate "bubbles" for sentences, within which we can grow circuit structure piecemeal. We will only express the rewrite rules; the generators of lower dimension are implicit. We aim to recover the linear ordering of words in text (essential to any syntax) by traversing the top surface of a chain of bubbles representing sentence structure in text – this order will be invariant up to compact closed isomorphisms. The diagrammatic consequence of these assumptions is that we will be working with a conservative generalisation of graph-rewriting defined by local rewriting rules. The major distinction is that locality can be redefined up to homotopy, which allows locally-defined rules to operate in what would be a nonlocal fashion in terms of graph neighbourhoods, as in Figure 2. The minor distinction is that rewrite rules are sensitive to twists in wires and

the radial order in which wires emanate from nodes, though it is easy to see how these distinctions can be circumvented by additional by imposing the equivalent of commutativity relations as bidirectional rewrites. It is worth remarking that one can devise weak n-categorical signatures to simulate turing machines, where output strings are e.g. 0-cells on a selected 1-cell, so rewrite systems of the kind we propose here are almost certainly expressively sufficient for anything; the real benefit is the interpretable geometric intuitions of the diagrams.

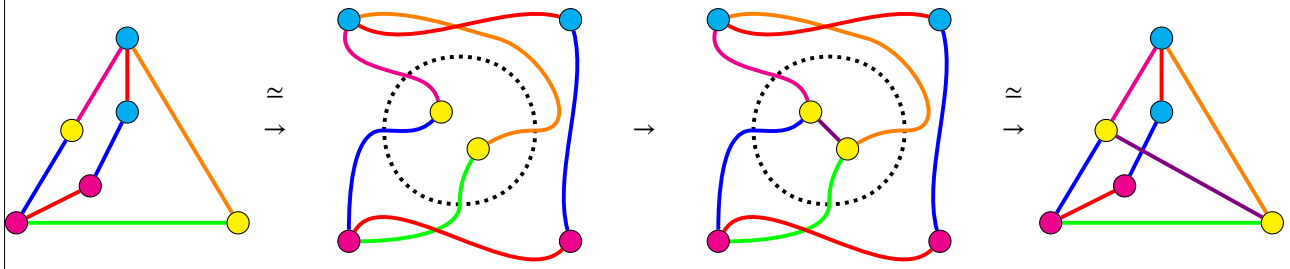
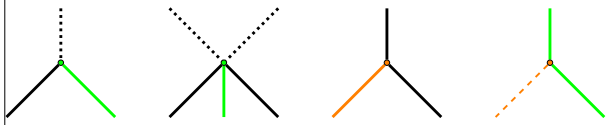


Figure 2: In this toy example, obtaining the same rewrite that connects the two yellow nodes with a purple wire using only graph-theoretically-local rewrites could potentially require an infinite family of rules for all possible configurations of pink and cyan nodes that separate the yellow, or would otherwise require disturbing other nodes in the rewrite process. In our setting, strong compact closure homotopies handle navigation between different spatial presentations so that a single rewrite rule suffices: the source and target notated by dotted-black circles. Despite the expressive economy and power of finitely presented signatures, we cannot "computationally cheat" graph isomorphism: formally we must supply the compact-closure homotopies as part of the rewrite, absorbed and hidden here by the $\simeq$ notation.

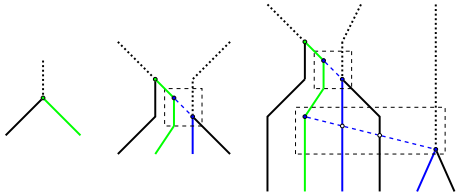
THE PLAN: We start with simple sentences that only contain a single intransitive or transitive verb. Then we consider more general sentences. For these two steps, we characterise the expressive capacity of our rules in terms of a context-sensitive grammar that corresponds to the surface structure of the derivations. Then we introduce text structure as lists of sentences with coreferential structure on nouns, along with a mathematical characterisation of coreferential structure and a completeness result of our rules with respect to them. Then we (re)state and prove the text circuit theorem: that the fragment of language we have built with the syntax surjects onto text circuits. Finally we examine how we may model extensions to the expressive capacity of text circuits by introduction of new rewrite rules.

Definition 0.1.2 (CSG for simple sentences)**.** We may gauge the expressivity of simple sentences with the following context sensitive grammar.

For verbs, adjectives, and modifiers, depicted unsaturated nouns as dotted and saturated with solid black lines, we have:

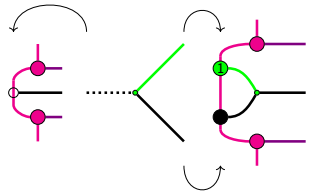


Adpositions require several helper-generators; we depict for example the beginning of the sequence of derivations that result from appending adpositions to an intransitive verb (the generators are implicit in the derivations):



Proposition 0.1.3. Up to labels, the simple-sentence rules yield the same simple sentences as the CSG for simple sentences.

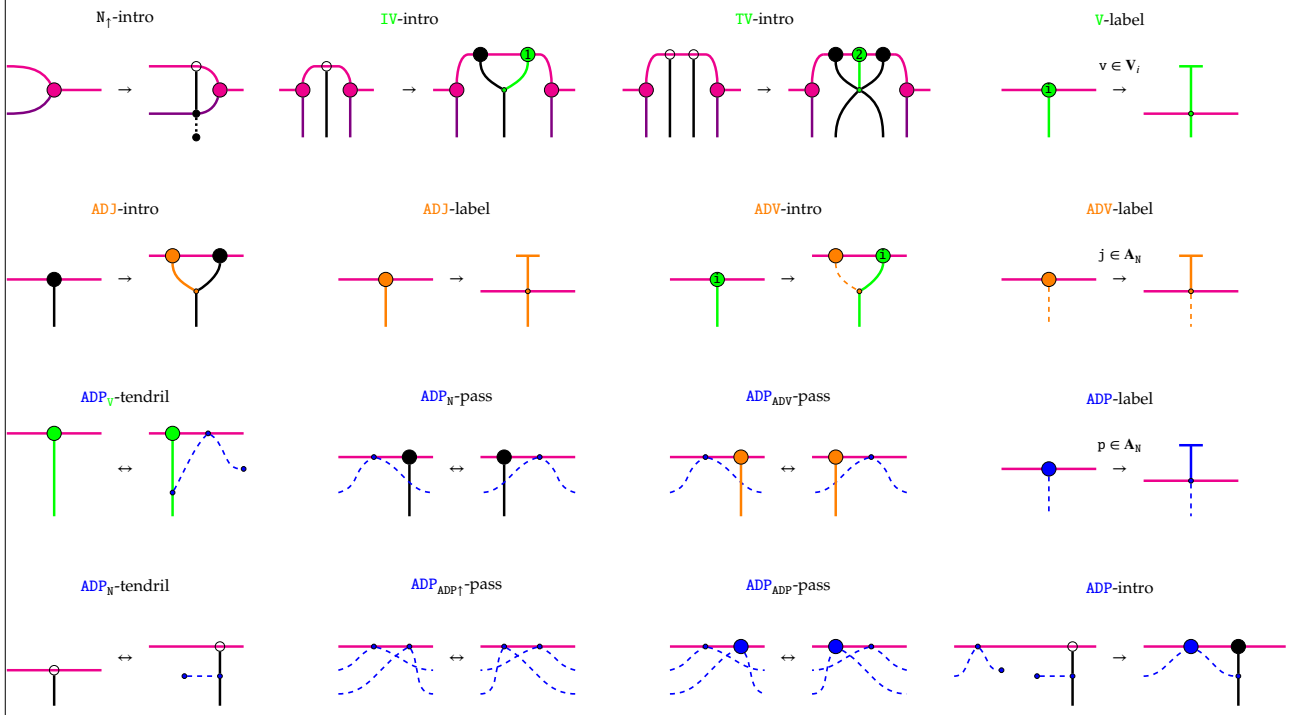
Proof. By graphical correspondence; viewing nodes on the pink surface as 1-cells, each rewrite rule yields a 2-cell. For example, for the IV-intro:



□

0.1.2 Simple sentences

Simple sentences are sentences that only contain a single intransitive or transitive verb. Simple sentences will contain at least one noun, and may optionally contain adjectives, adverbs, and adpositions. The rules for generating simple sentences are as follows:



The $\mathtt{N}_\uparrow$-intro rule introduces new unsaturated nouns from the end of a simple sentence. The IV-intro rule applies when there is precisely one unsaturated noun in the sentence, and the TV-intro rule applies when there are precisely two. Both verb-introduction rules saturate their respective nouns, which we depict with a black bulb. Adjectives may be introduced immediately preceding saturated nouns, and adverbs may be introduced immediately preceding any kind of verb. The position of adpositions in English is context-sensitive. To capture this, the $\mathtt{ADP}_\mathtt{V}$-tendril rule allows an unsaturated adposition to appear immediately after a verb; a bulb may travel by homotopy to the right, seeking an unsaturated noun. Conversely, the bidirectional $\mathtt{ADP}_\mathtt{N}$-tendril rule sends a mycelic tendril to the left, seeking a verb. The two pass-rules allow unsaturated adpositions to swap past saturated nouns and adjectives; note that by construction, neither verbs nor adverbs will appear in a simple sentence to the right of a verb, so unsaturated adpositions will move right until encountering an unsaturated noun. In case it doesn't, the tendril- and pass- rules are bidirectional and reversible.

0.1.3 *Complex sentences*

Now we consider two refinements; conjunctions, and verbs that take sentential complements. we may have two sentences joined by a conjunction, e.g. `Alice dances while Bob drinks`. We may also have verbs that take a sentential complement rather than a noun phrase, e.g. `Alice sees Bob dance`; these verbs require nouns, which we depict as wires spanning bubbles.

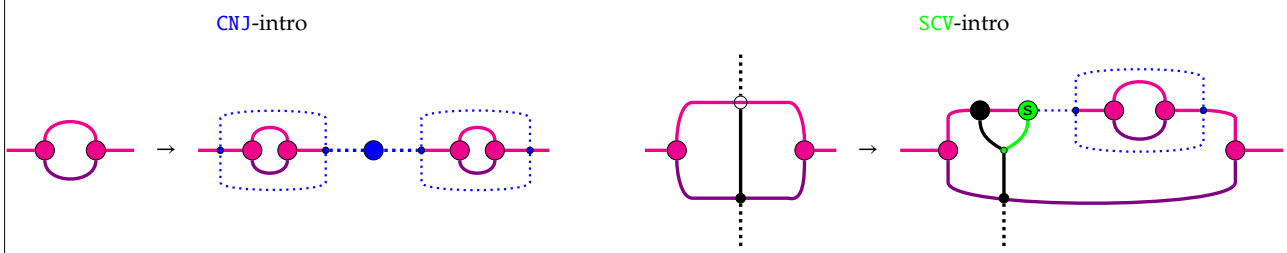
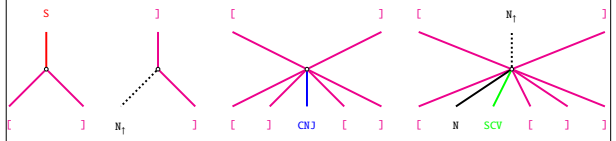


Figure 3: The dotted-blue wires do not contentfully interact with anything else, but this noninteraction disallows overgeneration cases where adpositional phrases might interject between SCV verbs and their sentential complement, e.g. `Alice sees at lunch Bob drink`. The dotted-blue wires also indicate a diagrammatic strategy for extensions to accommodate noun phrases, to be explored later.

Definition 0.1.4 (Sentence structure)**.** A sentence can be:

- a simple sentence, which...
- ... may generate unsaturated nouns from the right.
- a pair of sentences with a conjunction in between.
- (if there is a single unsaturated noun) a sentence with a sentential-complement verb that scopes over a sentence.

As a CSG, these considerations are respectively depicted as:



Proposition 0.1.5. Up to labels, the rules so far yield the same sentences as the combined CSG of Definitions 0.1.2 and 0.1.4.

Proof. Same correspondence as Proposition 0.1.3, ignoring the dotted-blue guards. □

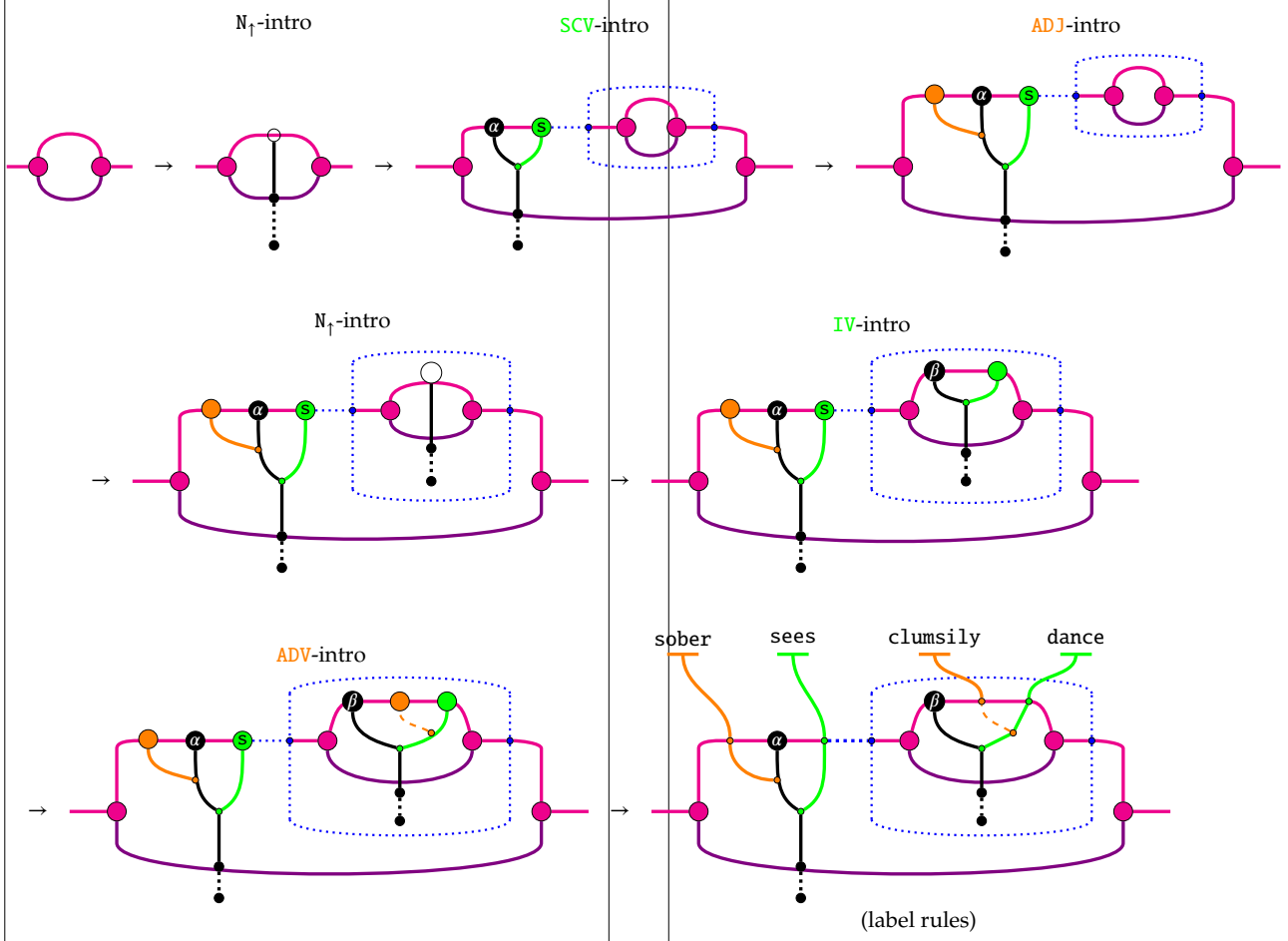


Figure 4:

Example 0.1.6 (`sober` α `sees drunk` β `clumsily dance.`). Now we can see our rewrites in action for sentences. As a matter of convention – reflected in how the various pass- rules do not interact with labels – we assume that labelling occurs after all of the words are saturated. We have still not introduced rules for labelling nouns: we delay their consideration until we have settled coreferential structure. For now they are labelled informally with greeks.

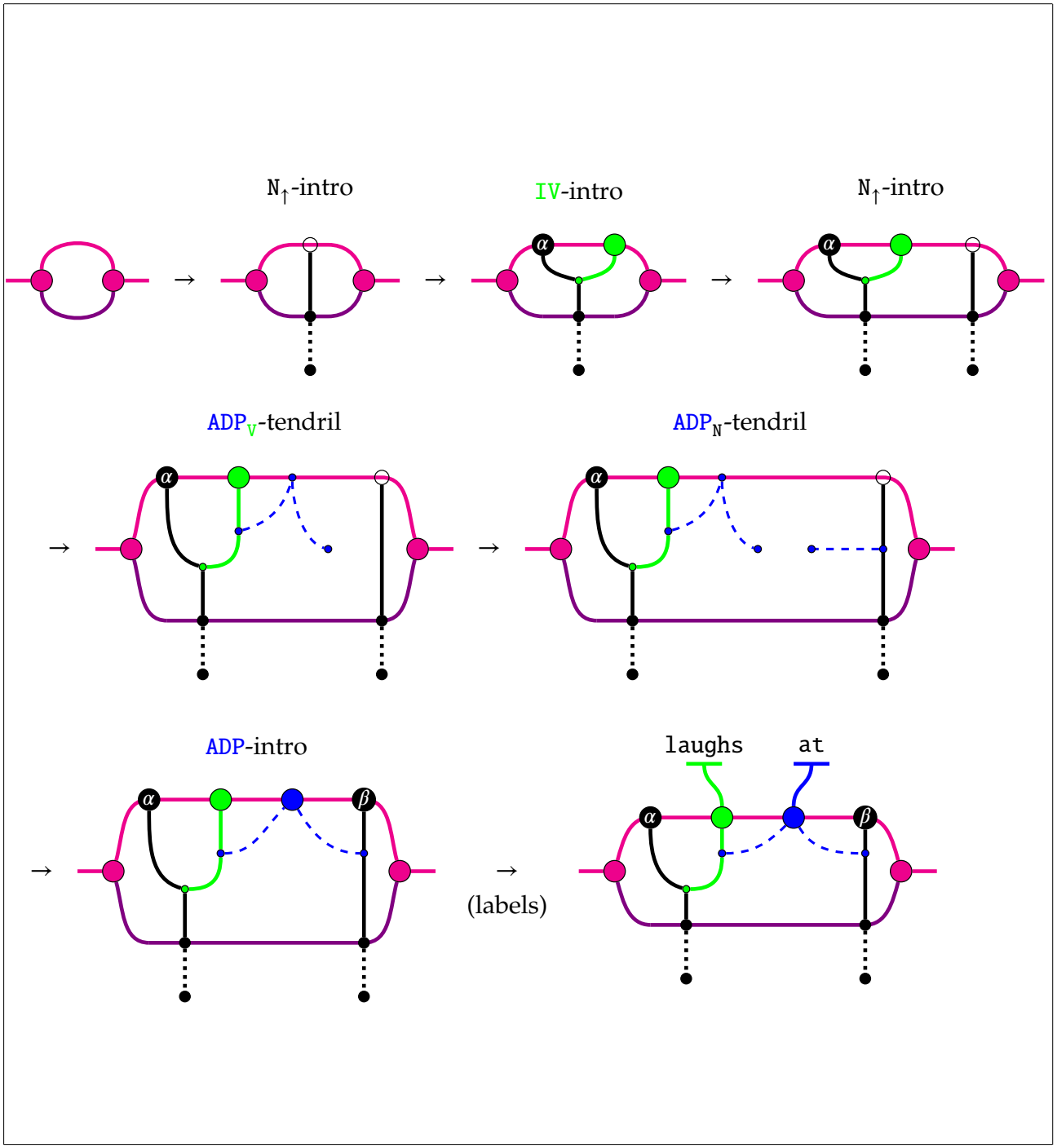


Figure 5:

Example 0.1.7 (α `laughs at` β)**.** Adpositions form by first sprouting and connecting tendrils under the surface. Because the tendril- and pass- rules are bidirectional, extraneous tendrils can always be retracted, and failed attempts for verbs to find an adpositional unsaturated noun argument can be undone. Though this seems computationally wasteful, it is commonplace in generative grammars to have the grammar overgenerate and later define the set of sentences by restriction, which is reasonable so long as computing the restriction is not computationally hard. In our case, observe that once a verb has been introduced and its argument nouns have been saturated, only the introduction of adpositions can saturate additionally introduced unsaturated nouns. Therefore we may define the finished sentences of the circuit-growing grammar to be those that e.g. contain no unsaturated nodes on the surface, which is a very plausible linear-time check by traversing the surface.

0.1.4 *Text structure and noun-coreference*

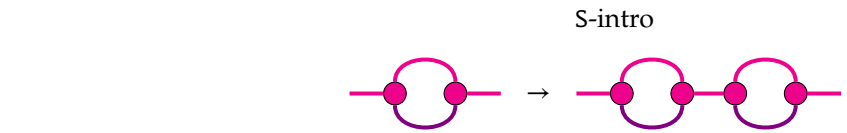


Figure 6: Only considering words, text is just a list of sentences. However, for our purposes, text additionally has *coreferential structure*. Ideally, we would like to connect "the same noun" from distinct sentences as we would circuits.

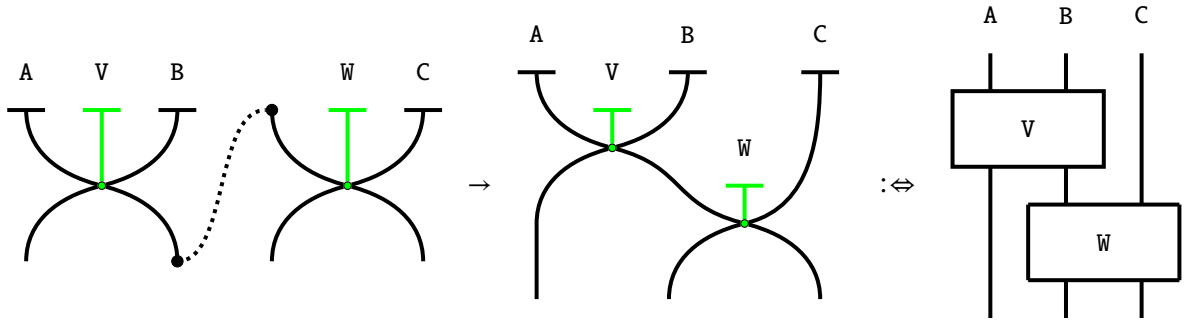


Figure 7: We choose the convention of connecting from left-to-right and from bottom-to-top, so that we might read circuits as we would text: the components corresponding to words will be arranged left-to-right and top-to-bottom. Connecting nouns across distinct sentences presents no issue, but a complication arises when connecting nouns within the same sentence as with reflexive pronouns e.g. `Alice likes herself`.

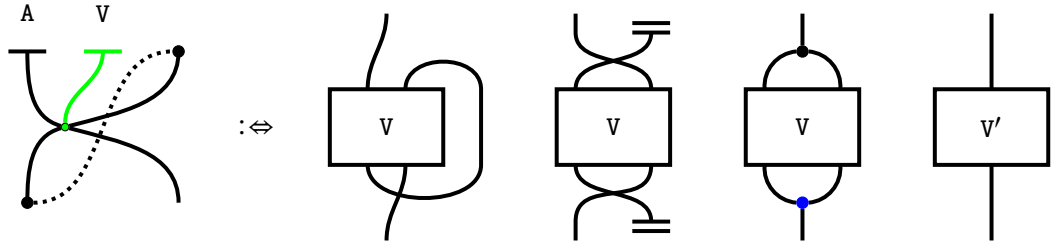
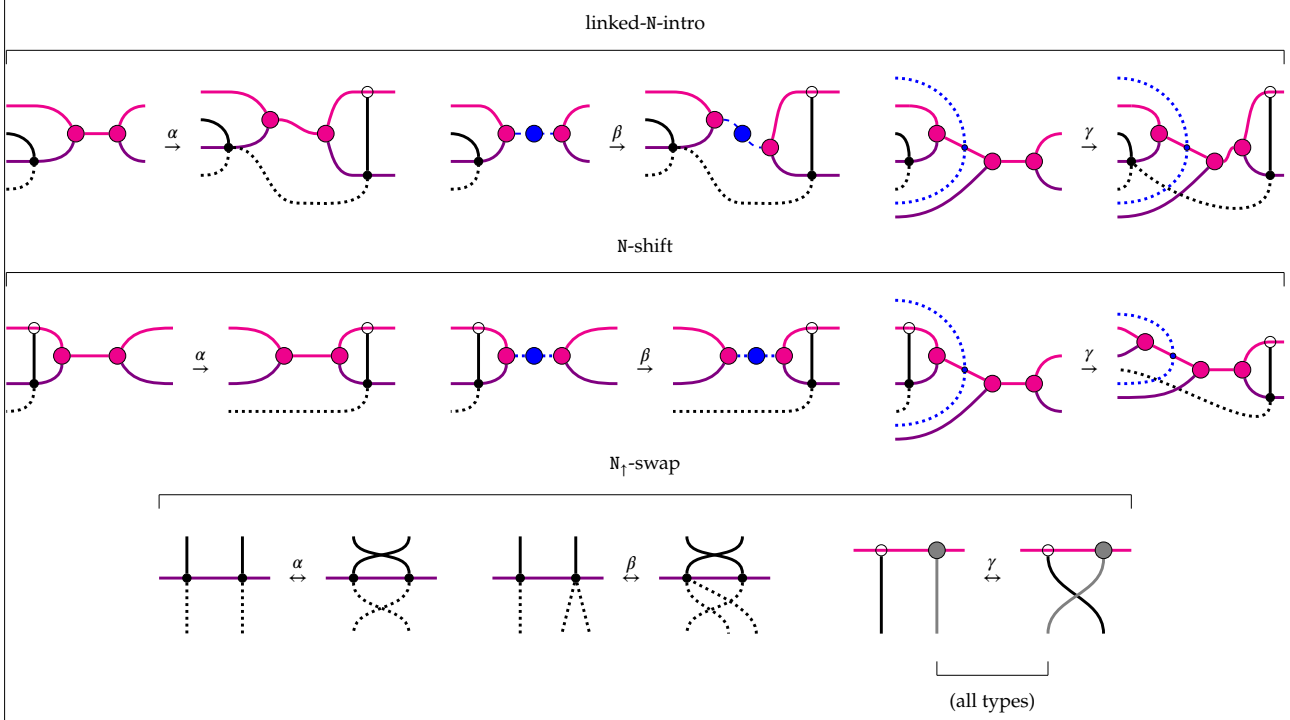


Figure 8: Reflexive coreference would violate of the processivity condition of string diagrams for symmetric monoidal categories. Not all symmetric monoidal categories possess the appropriate structure to interpret such reflexive pronouns, but there exist interpretative options. From left to right in roughly decreasing stringency, compact closed categories are the most direct solution. More weakly, traced symmetric monoidal categories also suffice. If there are no traces, so long as the noun wire possesses a monoid and comonoid, a convolution works. If all else fails, one can just specify a new gate. We will define coreference structure to exclude such reflexive coreference and revisit the issue as an extension.

Now we will deal with coreferential structure and noun-labels. TODO: need more linked-intro variants to handle leaving right-side of conjunction, nested SCV, CNJ to SCV, and SCV to CNJ. Consider using the same graywire convention to simplify case analysis?



The linked-N-intro rules introduce a new unsaturated noun in the next sentence that coreferences the noun in the previous sentence that generated it. The N-shift rules allow any unsaturated noun to move into the next sentence. For both of the previous rules, the β variant handles the case where the next sentence is related to the first by a conjunction. Observe that nouns with a forward coreference have two dotted-black wires leaving the root of their wires, which distinguishes them from nouns that only have a backward coreference or no coreference at all, which only have a single dotted-black wire leaving the root of their wire.

The N-swap rule variants allow a unsaturated noun with no forward coreferences to swap places with any unsaturated noun that immediately succeeds it.

When the structure of coreferences is set, we propagate noun labels from the head of each list. The rules for noun-label propagation are as follows:

Example 0.1.8 (`sober Alice sees Bob clumsily dance.  She laughs at him.`).

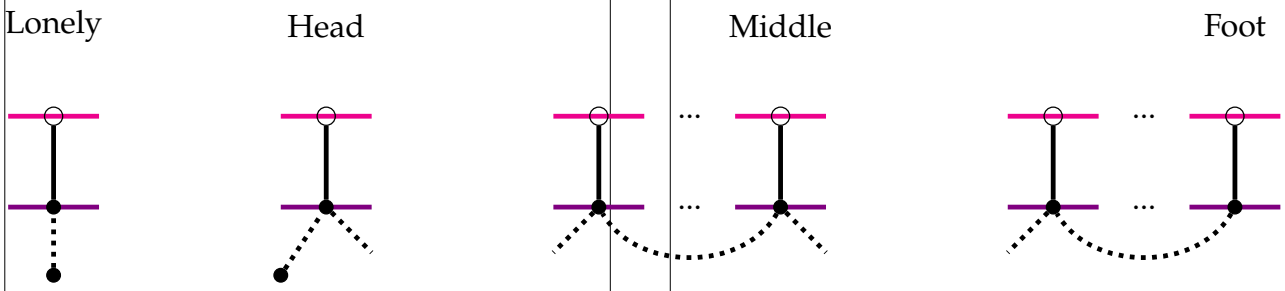


Figure 9: At this point, it is worth establishing some terminology about the kinds of unsaturated nouns we have in play. The kinds of nouns are distinguished by their tails. *Lonely* nouns have no coreferences, their tails connect to nothing. *Head* nouns have a forward coreference in text; they have two tails, one that connects to nothing and the other to a noun later in text. *Middle* nouns have a forward and backward coreference; they have two tails, one that connects to a noun in some preceding sentence, and one that connects forward to a noun in a succeeding sentence. *Foot* nouns only have a backward coreference; they have a single tail connecting to a noun in some preceding sentence.

0.1.5 Text circuit theorem

Definition 0.1.9 (Text Circuits)**.** *Text circuits* are made up of three ingredients:

- wires
- boxes, or gates
- boxes with holes that fit a box, or 2nd order gates

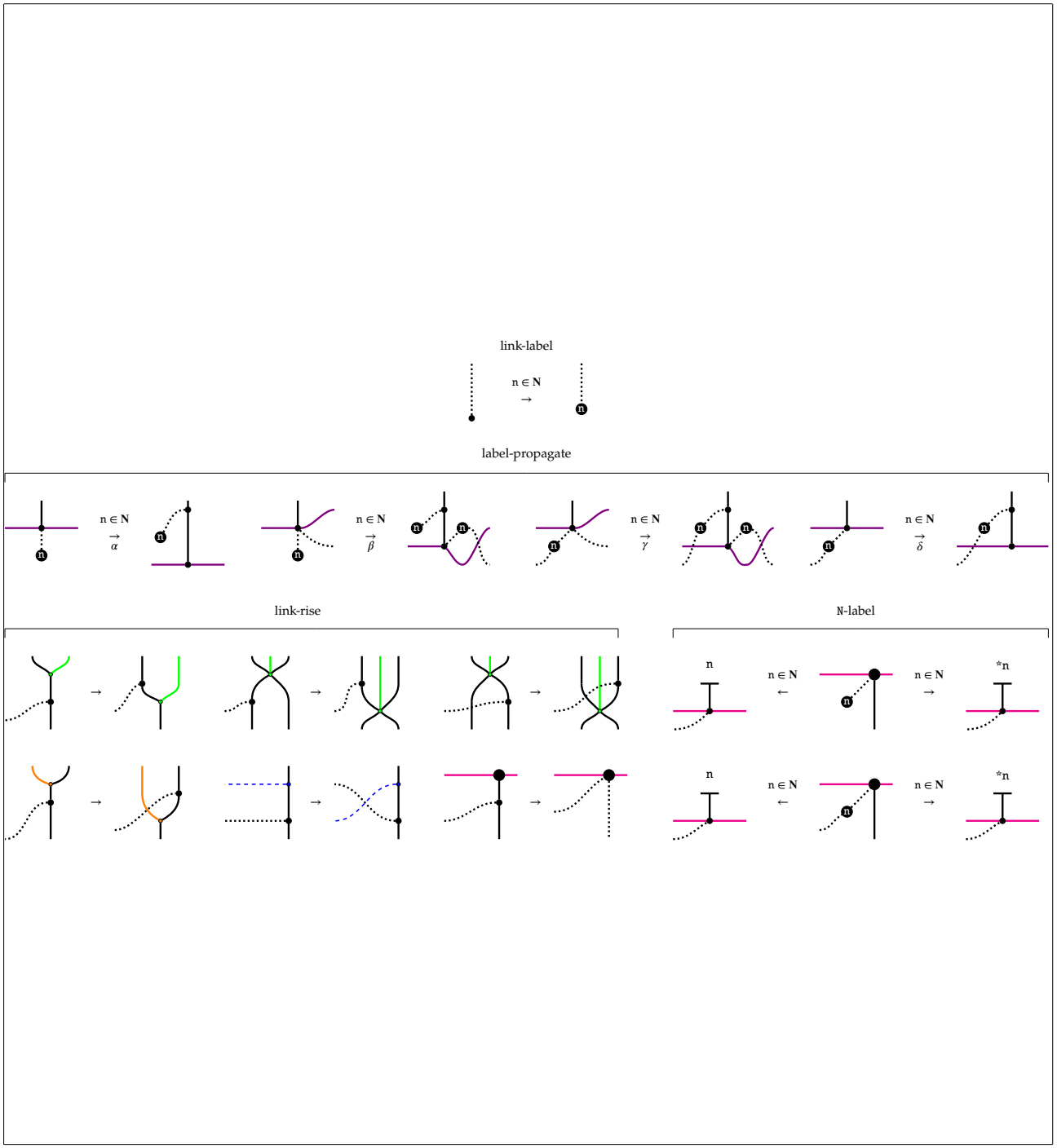


Figure 10: The $n \in \mathbf{N}$ notation indicates a family of rewrites (and generators) for each noun in the lexicon. Link-label assigns a noun to a diagrammatically linked collection of coreferent nouns, and link-propagation is a case analysis that copies a link label and distributes is across coreferent nouns. Link-rise is a case analysis to connect labels to the surface, and finally N-label allows a saturated noun to inherit the label of its coreference class, which may either be a noun $\mathbf{n}$ or a pronoun appropriate for the noun, notated ${}^{*}\mathbf{n}$

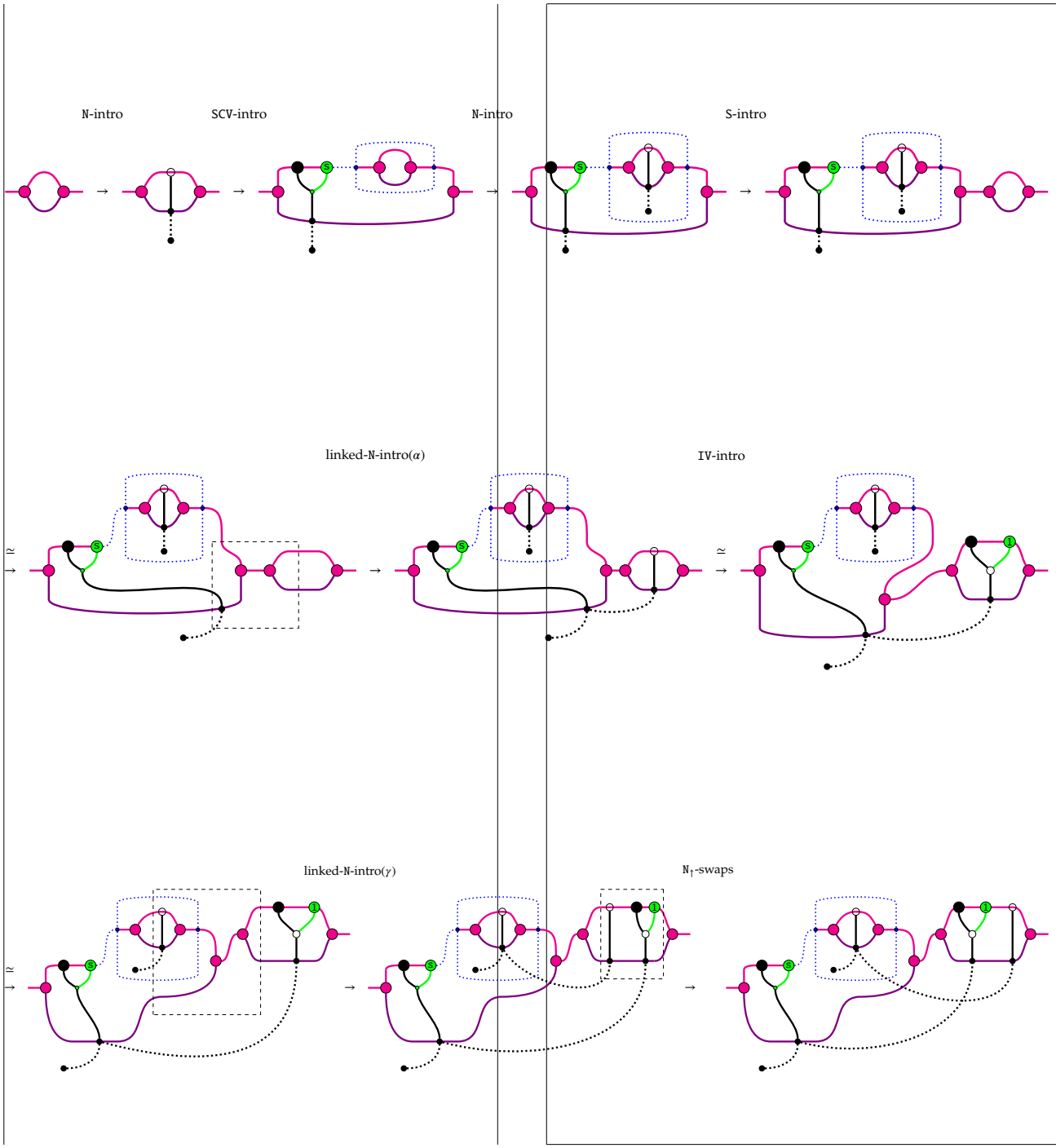


Figure 11: We start the derivation by setting up the sentence structure using S- and SCV-intro rules, and two instances of N-intro, one for Alice, and one for Bob. Observe how the N-intro for Bob occurs within the sub-sentence scoped over by the SCV-rule.

Figure 12: By homotopy, we can rearrange the previous diagram to obtain the source of the linked-N-intro rewrite in the dashed-box visual aid. Observe how we drag in the root of what is to be Alice's wire. Then we use the IV-intro in the second sentence, which sets up the surface structure `she laughs`, and the deep structure for bookkeeping that `she` refers to `Alice`.

Figure 13: By homotopy again, we can do the same for Bob, this time setting up for the γ variant of linked-N-intro which handles the case when the spawning noun is within the scope of an SCV. Then by applying a series of $\mathtt{N}_{\uparrow}$-swaps, the unsaturated noun is placed to the right of the intransitive verb phrase.

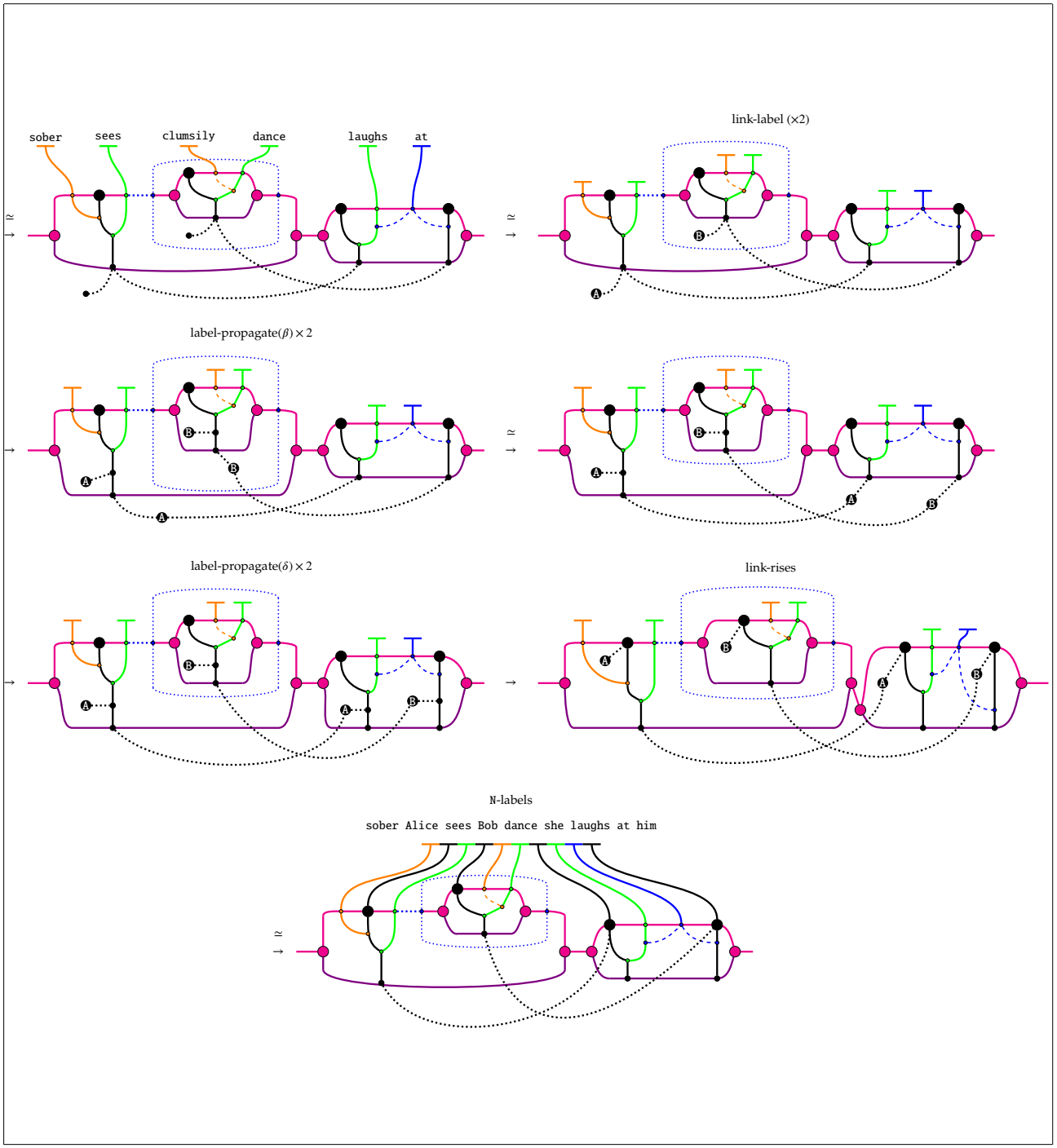


Figure 14: We've already done the surface derivation for the two sentences separately in Figures 4 and 5; since neither of those derivations touch the roots of noun-wires, we can emulate those derivations and skip ahead to the first diagram.

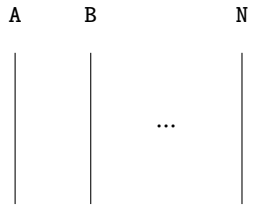


Figure 15: Nouns are represented by wires, each 'distinct' noun having its own wire.

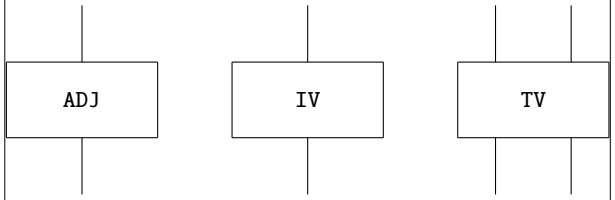


Figure 16: We represent adjectives, intransitive verbs, and transitive verbs by gates acting on noun-wires. Since a transitive verb has both a subject and an object noun, that will then be two noun-wires, while adjectives and intransitive verbs only have one.

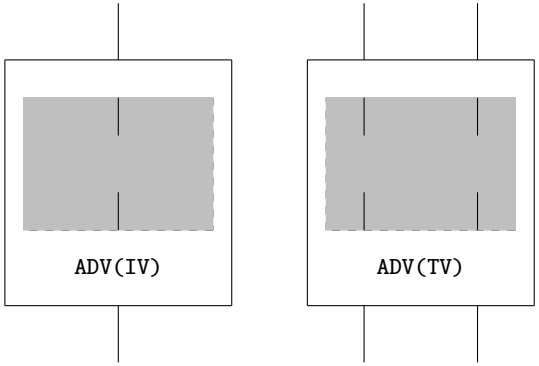


Figure 17: Adverbs, which modify verbs, we represent as boxes with holes in them, with a number of dangling wires in the hole indicating the shape of gate expected, and these should match the input- and output-wires of the box with the whole.

Now we demonstrate that *finished* text diagrams yield unique text circuits up to homotopy. Text circuits are presented again in the margins as a reminder. The strategy will be to present new rewrites that transform coreferential structure into symmetric monoidal wire-connectivity.

Definition 0.1.10 (Finished text diagram)**.** The circuit-growing grammar produces *text diagrams*. We call a text diagram *finished* if all surface nodes are labelled.

Proposition 0.1.11. Finished text diagrams yield text, up to interpreting distinct sentences as concatenated with punctuation `.`, `,`, contentless conjunctions or complementisers – such as `and`, or `that` respectively.

Proof. Sentence-wise grammaticality is gauged by Propositions 0.1.3 and 0.1.5. When multiple sentences occur within the scope of a `SCV` we might prefer the use of contentless complementisers and conjunctions, e.g. `Alice sees( Bob draws Charlie drinks ) Dennis dances` is grammatically preferable but meaningfully equivalent to `Alice sees that Bob draws and Charlie drinks , and Dennis dances .` For our purposes it makes no difference whether surface text has these decorations, as the deep structure of text diagrams encodes all the information we care to know. □

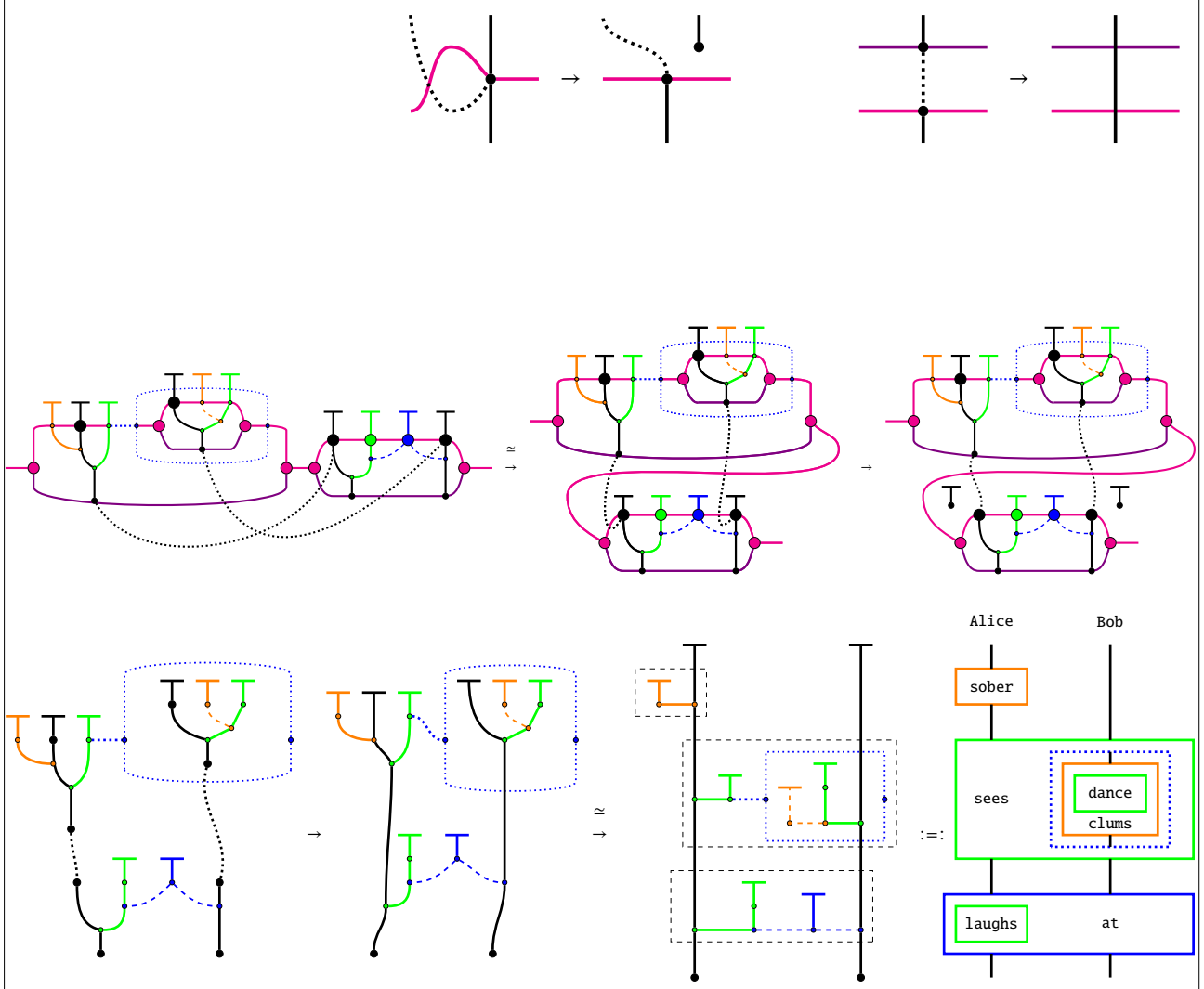
Lemma 0.1.12. The unsaturated noun kinds listed in Figure 9 are exhaustive, hence nouns that share a coreference are organised as a diagrammatic linked-list.

Proof. The `N`-intro rule creates lonely nouns. Head nouns can only be created by the linked-`N`-intro applied to a lonely noun. Any new noun created by linked-`N`-intro is a foot noun. The linked-`N`-intro rule turns foot nouns into middle nouns. These two intro- rules are the only ones that introduce unsaturated nouns, so it remains to demonstrate that no other rules can introduce noun-kinds that fall outside our taxonomy. The `N`-shift rule changes relative position of either a lonely or foot noun but cannot change its kind. The `N`-swap rule may start with either a lonely or foot noun on the left and either a head or middle noun on the right, but the outcome of the rule cannot change the starting kinds as tail-arity is conserved and the local nature of rewrites cannot affect the ends of tails. □

Lemma 0.1.13. No nouns within the same sentence are coreferentially linked.

Proof. Novel linked nouns can only be obtained from the linked-`N`-intro rule, which places them in succeeding sentences. The swap rules only operate within the same sentence and keep the claim invariant. The `N`-shift rules only apply to nouns with no forward coreferences; nouns with both forward and backward coreferences cannot leave the sentence they are in. Moreover, `N`-shift is unidirectional and only allows the rightmost coreference in a linked-list structure to move to later sentences. So there is no danger of an `N`-shift breaking the invariant. □

Construction 0.1.14 (Text to circuit).



Proposition 0.1.15 (Finished text diagrams yield unique-up-to-processive-isotopy text circuits). *Proof.* Every sentence corresponds to a gate up to notation, and we have a handle on sentences via Propositions 0.1.3 and 0.1.5. Lemmas 0.1.12 and 0.1.13 guarantee processivity. Uniqueness-up-to-processive-isotopy is inherited: text diagrams themselves are already specified up-to-connectivity, which is strictly more general than processive isotopy. Therefore, for any circuit C obtained from a text diagram T by Construction 0.1.14, T can be

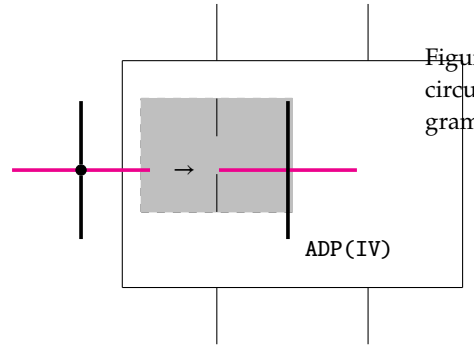


Figure 18: Similarly, adpositions also modify verbs, by moreover adding another noun-wire to the right.

Figure 19: For verbs that take sentential complements and conjunctions, we have families of boxes to accommodate input circuits of all sizes. They add another noun-wire to the left of a circuit.

Figure 23: We turn finished text diagrams into text circuits by operating *in* grammatical system th

Figure 24: In the first step, by Lemmas 0.1.12 and 0.1.13, we can always rearrange a finished text diagram such that the noun wires are processive.

In the second step, use the first rewrite of Construction 0.1.14 to prepare the wires for connection.

In the third step, we just ignore the existence of the bubble-scaffolding and the loose scalars. We could in principle add more rewrites to melt the scaffolding away if we wanted, but who cares? ...

In the fourth step, we apply the second and third rewrites of Construction 0.1.14 to connect the wires and eliminate nodules underneath labels. We can also straighten up the wires a bit and make them look proper.

At this point, we're actually done, because the resulting diagram *is already a text circuit up to a choice of notation*.

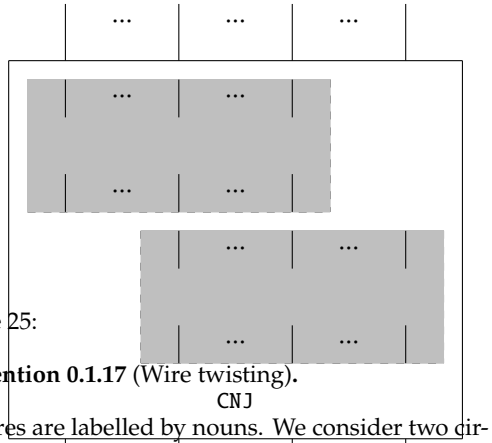


Figure 20: Conjunctions are boxes that take two circuits which might share labels on some wires.

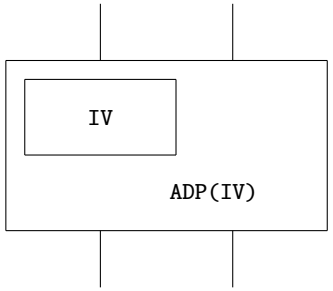


Figure 21: Of course filled up boxes are just gates.

Figure 25:

Convention 0.1.17 (Wire twisting)**.**

Wires are labelled by nouns. We consider two circuits the same if their gate-connectivity is the same. In particular, this means that we can eliminate unnecessary twists in wires to obtain diagrammatically simpler representations.

Figure 26:

Convention 0.1.18 (Sliding)**.**

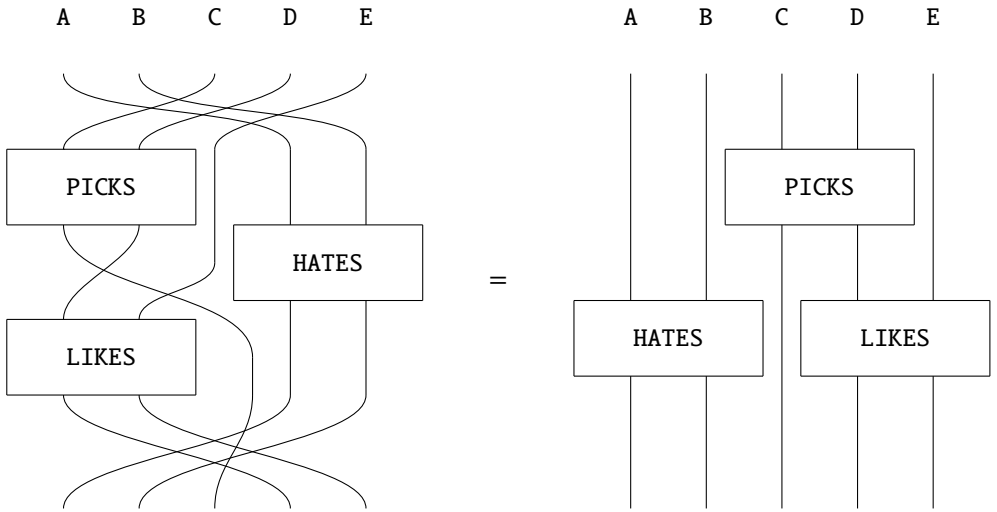
Since only gate-connectivity matters, we consider circuits the same if all that differs is the horizontal positioning of gates composed in parallel.

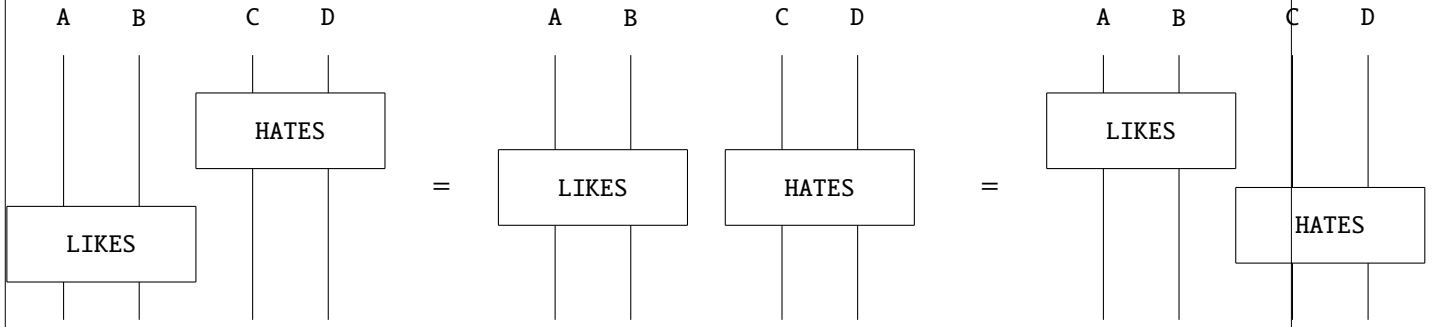
Convention 0.1.19 (Reading text circuits)**.**

Text circuits ought to be presented so that they can be read from top to bottom and from left to right, like English text.

modified up to processive-isotopy on noun wires to yield T' and another circuit C' that only differs from C up to processive isotopy, and all C' can be obtained in this way. □

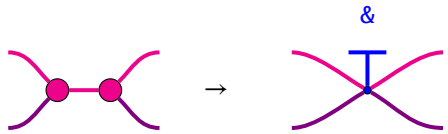
The converse of Proposition 0.1.15 would be that any text circuit that can be formed by the composition of symmetric monoidal categories and of plugging gates into boxes yields a text diagram. This would mean that text circuit composition is acceptable as a generative grammar for text. Establishing this converse requires elaboration of some conventions.



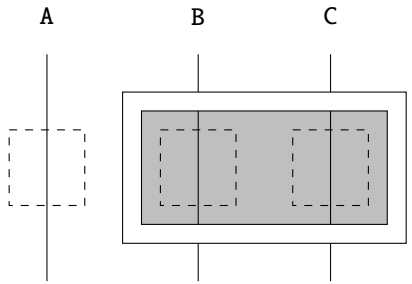


Convention 0.1.20 (Arbitary vs. fixed holes)**.** Diagrammatically, adverbs and adpositions are depicted with no gap between the bounding box and their contents, whereas conjunctions and verbs with sentential complement are depicted with a gap; this is a visual indication that the former are type-sensitive, and the latter can take any circuit.

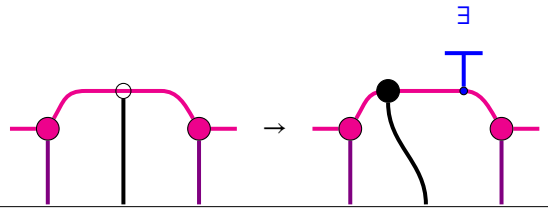
Convention 0.1.21 (Contentless conjunctions)**.** Conventions **??** and 26 require something else to allow them to work at the same time. The left text circuit may be read `C hates D. A likes B.`, and the right as `A likes B. C hates D.` The middle text circuit can be rewritten as either the left or right, which can be done if we're happy to make such choices. A choiceless approach requires something to distinguish the fact that the gates are parallel: a "contentless" conjunction, such as `and`, or `while`. In terms of text diagrams, we want a rewrites that introduces such contentless conjunctions and witnesses their associativity, like this:



Convention 0.1.22 (Lonely wires)**.** There's really only a single kind of text circuit we can draw that doesn't obviously correspond to a text diagram, and that's one where gates are missing.



In process theories, wires are identity processes that do nothing to their inputs. We require a text diagram analogue, and an intransitive "null-verb" in English that seems to work is `is`, in the sense of `exists`. In terms of text diagrams, we want a rewrite that introduces such contentless verbs, like this:



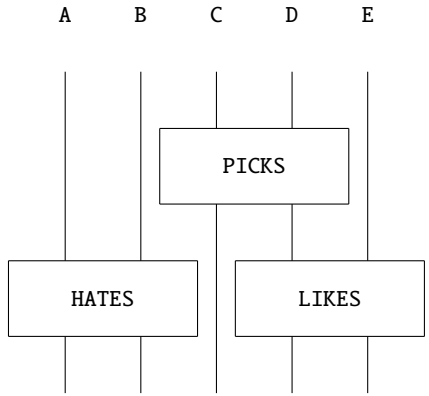


Figure 22: Gates compose sequentially by matching labels on some of their noun-wires and in parallel when they share no noun-wires, to give text circuits.

Remark 0.1.16. There are some oddities about our conventions that will make sense later when we consider semantics. For example, Convention 25 an acceptable thing to ask for syntactically but quite odd to think about at the semantic level, where we would like to think that distinct nouns manifest as different states on the same noun-wire-type. A semantic interpretation that makes use of this convention will become clearer in Section configspace . Similarly, Convention 0.1.22 wouldn't be true if we consider the order of text to reflect the chronological ordering of events, in which case there are implicit `...` `and then ...` conjunctions that distinguish ordered gates from parallel gates conjoined by an implicit `...` `while ...`; this particular complication is handled at the semantic level in Section statesactionmanner . The distinction in Convention 0.1.20 between typed and "untyped" higher-order processes will be given a suitable semantic interpretation in Section lassos .

Construction 0.1.23 (Circuit to text)**.** In the presence of additional rewrites from Conventions 0.1.21 and 0.1.22, every text circuit is obtainable from some text diagram, up to Conventions 25 and **??**.

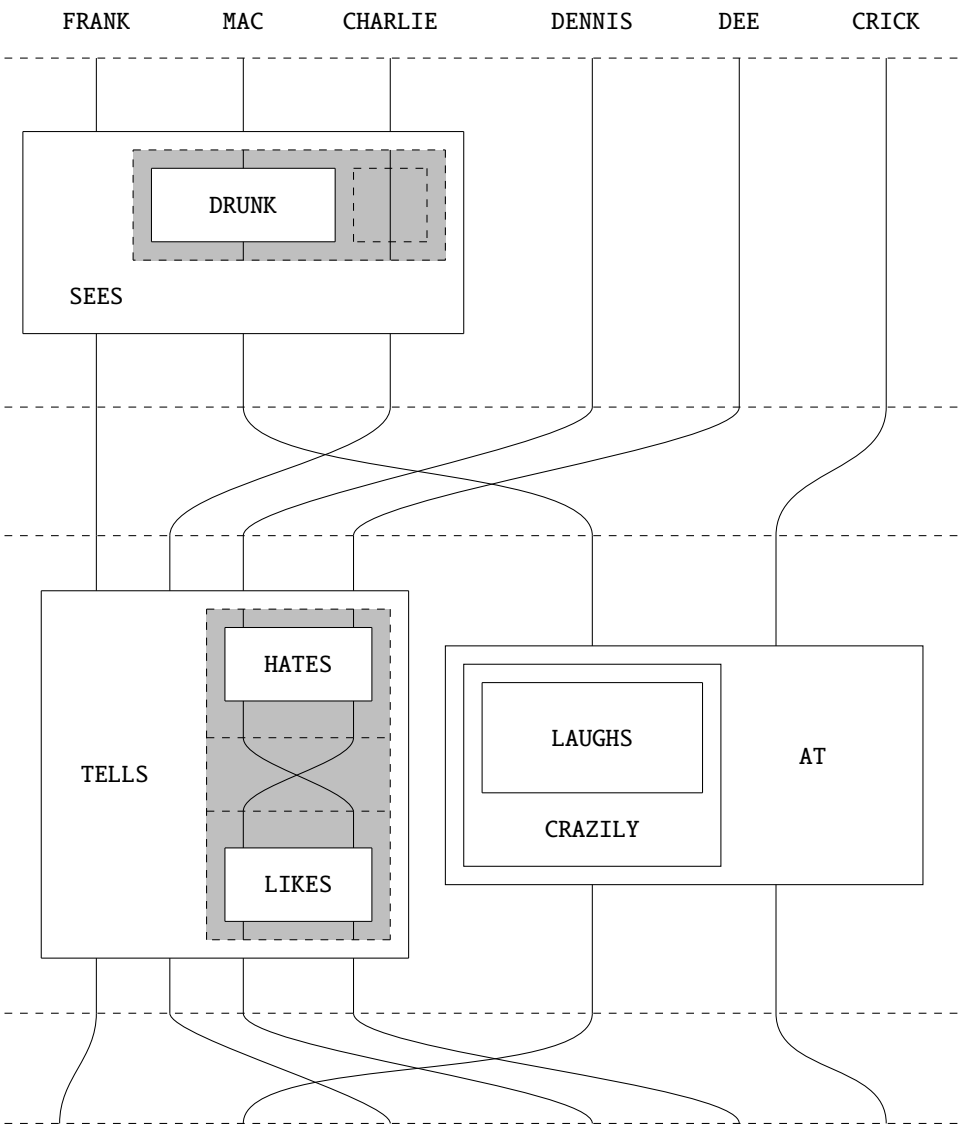


Figure 27: Starting with a circuit, we may use Convention 25 to arrange the circuit into alternating slices of twisting wires and (possibly tensored) circuits, and this arrangement recurses within boxes. Slices with multiple tensored gates will be treated using Convention 0.1.21. By convention 0.1.22, we decorate lonely wires with formal `exists` gates, as in the `Frank sees` box. Observe how verbs with sentential complement are depicted with grey gaps, whereas the adverb and adposition combination of `Mac crazily laughs at Cricket` is gapless, according to Convention 0.1.20.

Figure 28: We then linearise the slices, representing top-to-bottom composition as left-to-right. Twist layers are eliminated, replaced instead by dotted connections indicating processive connectivity. The dashed vertical line distinguishes slices. This step of the procedure always behaves well, guaranteed by Proposition 0.1.12. Noun wires that do not participate in earlier slices can be shifted right until the slice they are introduced.

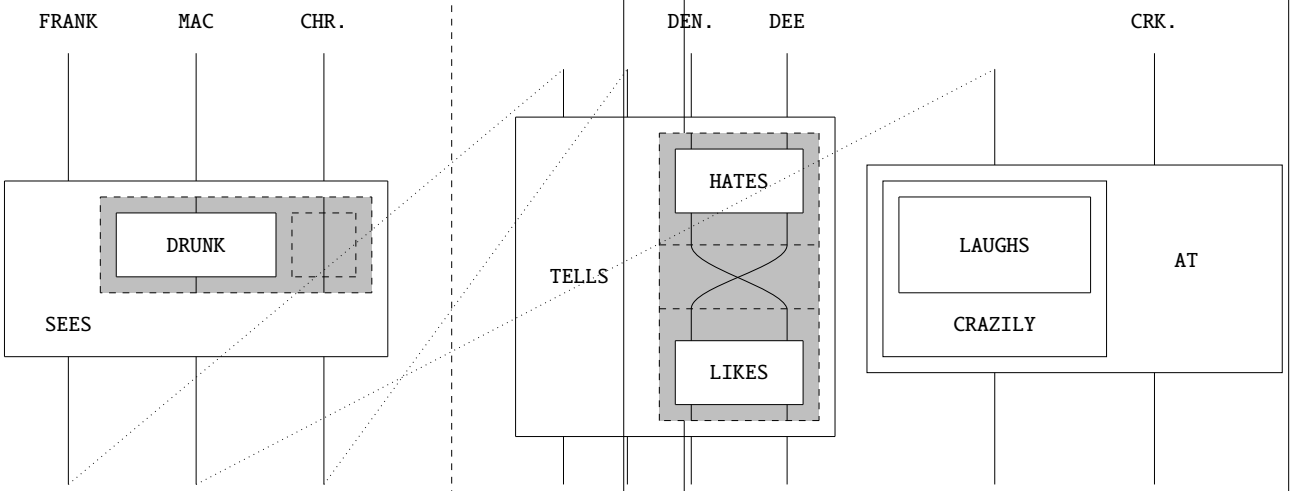
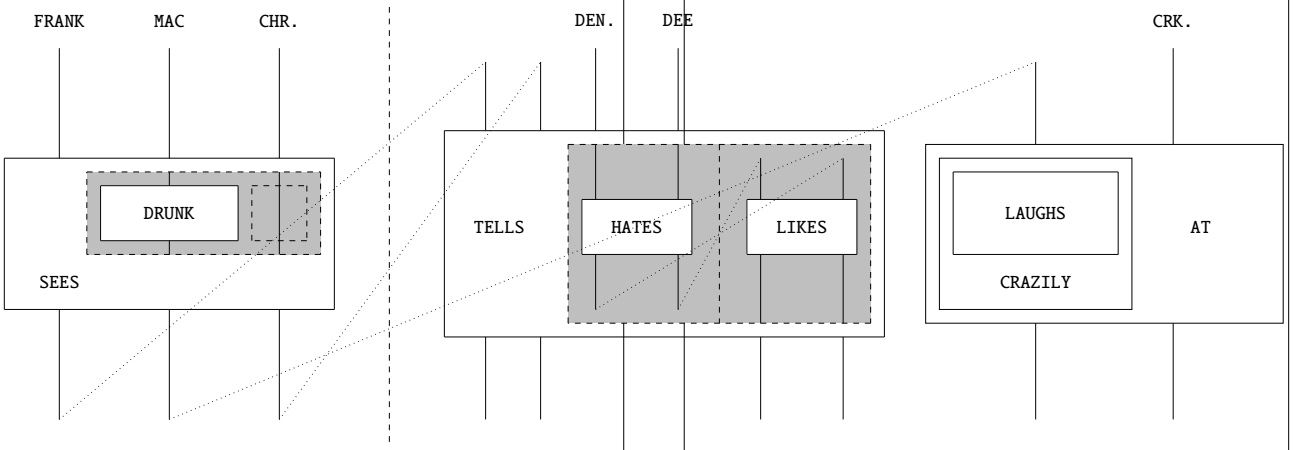


Figure 29: We recurse the linearisation procedure within boxes until there are no more sequentially composed gates. The linearisation procedure evidently terminates for finite text circuits. At this point, we have abstracted away connectivity data, and we are left with individual gates.



0.1.6 Extensions I: relative and reflexive pronouns

SUBJECT RELATIVE PRONOUNS

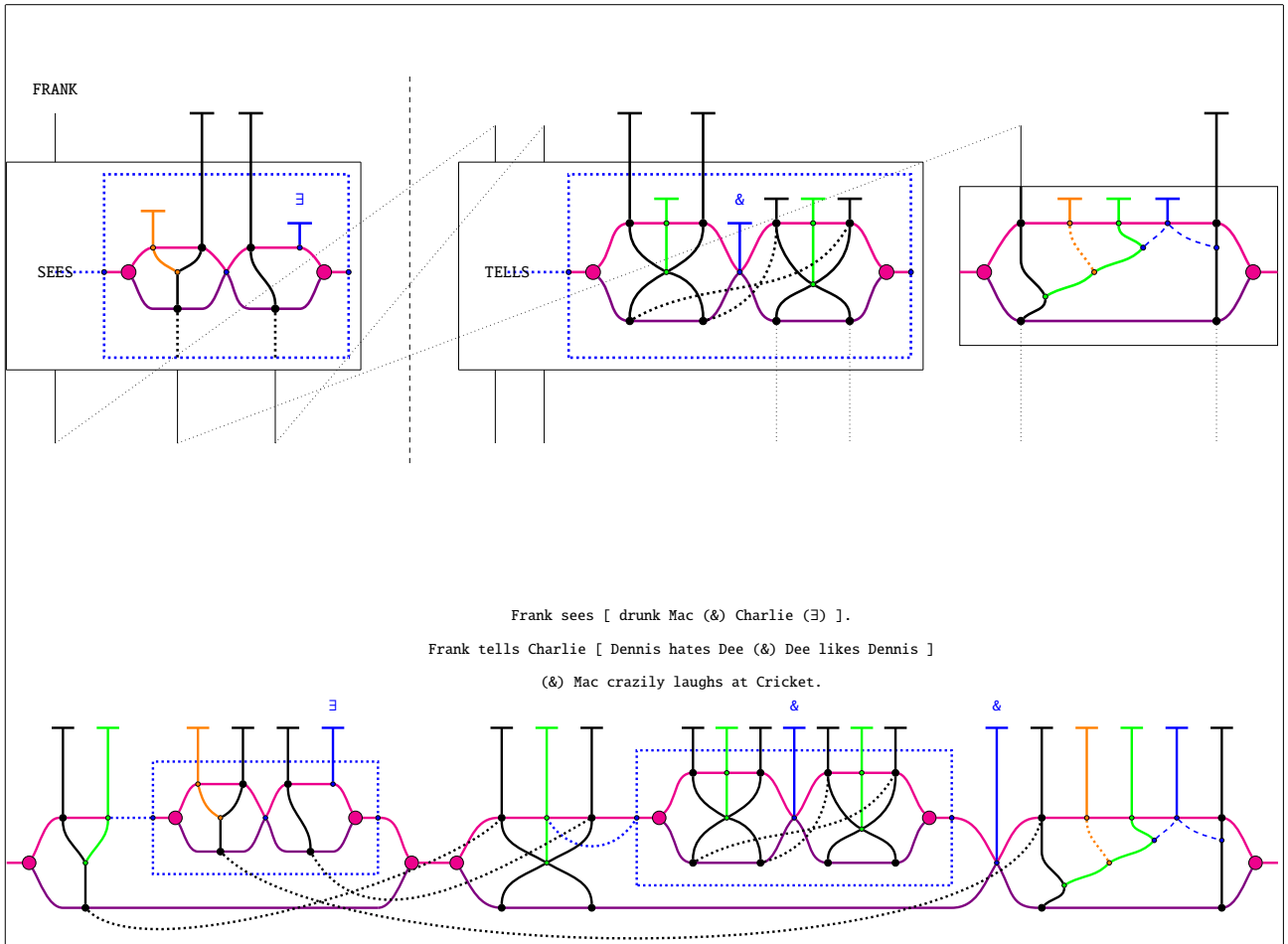


Figure 30: By Proposition 0.1.5, gates are equivalent to sentences up to notation, so we swap notations *in situ*. Conventions 0.1.21 and 0.1.22 handle the edge cases of parallel gates and lonely wires. Observe that the blue-dotted wiring in text diagrams delineates the contents of boxes that accept sentences.

Figure 31: Recursing notation swaps outwards and connecting left-to-right slices as sentence-bubbles connect yields a text circuit, up to the inclusion of rewrites from Conventions 0.1.21 and 0.1.22: applying the reverse of those rewrites and the reverse of text-diagram rewrites yields a valid text-diagram derivation, by Propositions 0.1.5 and 0.1.12.

Example 0.1.24.

Object relative pronouns

Example 0.1.25.

Reflexive pronouns

Example 0.1.26.

0.1.7 Extensions II: grammar equations

ATTRIBUTIVE VS. PREDICATIVE MODIFIERS

Example 0.1.27.

COPULAS

Example 0.1.28.

POSSESSIVE PRONOUNS

Example 0.1.29.

0.1.8 Extensions III: higher-order modifiers

INTENSIFIERS

Example 0.1.30.

COMPARATIVES

Example 0.1.31.

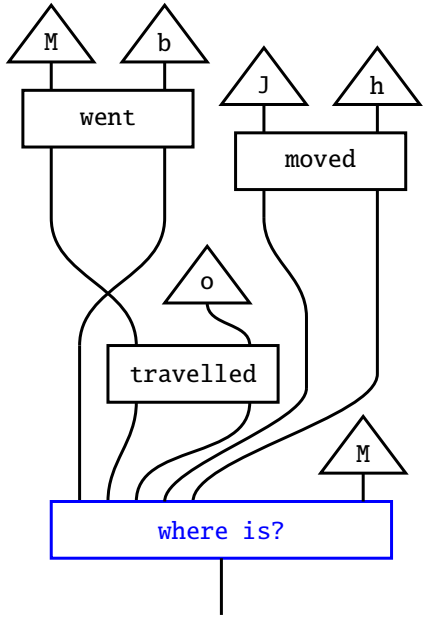
0.1.9 Equivalence to internal wirings

0.1.10 Related work

An example from Task 1, "single supporting fact", is:

```
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
(Query:) Where is Mary?
(Answer:) office.
```

Translating the setup of each task into a circuit of neural nets-to-be-learnt, and queries into appropriately typed measurements-to-be-learnt, each bAbi task becomes a training condition: the depicted composite process ought to be equal to the `office` state.



Solving bAbi in this way also means that each word gate has been learnt in a conceptually-compliant manner, insofar as the grounded meanings of words are reflected in how words interact and modify one another. One may argue that the verb `to go` and synonyms have been learnt-from-data in a way that coheres with human conceptions by appeal to the bAbi dataset.

0.2 Text circuits: details, demos, developments

This section covers some practical developments, conventions, references for technical details of text circuits. The most striking demonstration to date is that circuits are defined over a large enough fragment of language to *leverage* several bAbi tasks CITE , which are a family of 20 general-reasoning text tasks – the italicised choice of wording will be elaborated shortly. Each family of tasks consists of tuples of text in simple sentences concluded by a question, along with an answer. It was initially believed that world models were required for the solution of these tasks, but they have been solved using transformer architectures. While there is no improvement in capabilities by solving bAbi using text circuits, the bAbi tasks have been used as a dataset to learn word gates from data, in a conceptually compliant and compositional manner. Surprisingly, despite the low-data, low-compute regime, the tasks for which the current theory has the expressive capacity to cover are solved better by text circuits than by LSTMs; a proof-of-concept that with the aid of appropriate mathematics, not only might fundamental linguistic considerations help rather than hinder NLP, but also that explainability and capability are not mutually exclusive. Experimental details are elaborated in a forthcoming report CITE . While there are expressivity constraints contingent on theoretical development, this price buys a good amount of flexibility within the theoretically established domain: text circuits leave room for both learning-from-data and "hand-coded" logical constraints expressed process-theoretically, and naturally accommodate previously computed vector embeddings of words.

In practice, the process of obtaining transparently computable text goes through two phases. First, one has to obtain text circuits from text, which is conceptually simple: typelogical parsers for sentences can be modified to produce circuit-components rather than trees, and a separate pronomial resolution module dictates symmetric monoidal compositionality; details are in the same forthcoming report CITE . Second, one implements the text circuits on a computer. On quantum computers, boxes are modelled as quantum combs CITE . On classical computers, boxes are sandwiches of generic vector-manipulation neural nets, and boxes with 'dot dot dot' typing are interpreted as families of processes, which can be factored for instance as a pair of content-carrying gates along with a monoid+comonoid convolution to accommodate multiplicity of wires. The theoretical-to-practical upshot of text circuits when compared to DisCoCat is that the full gamut of compositional techniques, variations, and implementation substrates of symmetric monoidal categories may be used for modelling, compared to the restrictions inherent in hypergraph and strongly compact closed categories.

In terms of underpinning mathematical theory, the 'dot dot dot' notation within boxes is graphically formal (**?**), and interpretations of such boxes were earlier formalised in (**???**). The two forms of interacting composition, one symmetric monoidal and the other by nesting is elsewhere called *produoidal*, and the reader is referred to CITE for formal treatment and a coherence theorem. Boxes with holes may be interpreted in several different ways. Firstly, boxes may be considered syntactic sugar for higher-order processes in monoidal closed categories, and boxes are diagrammatically preferable to combs in this regard, since the latter admits

a typing pathology where two mutually facing combs interlock. Secondly, boxes need not be decomposable as processes native to the base category, admitting for instance an interpretation as elementwise inversion in linear maps, which specialises in the case of **Rel** (viewed as **Vect** over the boolean ring) to negation-by-complement. In some sense, none of these formalities really matter, on the view that text circuits are algebraic jazz for interpreting text, where facets are open to interpretation and modification.