STRING DIAGRAMS FOR TEXT

VINCENT WANG-MAŚCIANICA

ST. CATHERINE'S COLLEGE
THE UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
2023

Contents

(Acknowledgements will go in a margin note here.)

1
Text circuits for syntax

We want to establish that there is a systematic correspondence between text circuits and grammatically acceptable text, which would allow us to use text circuits as a generative grammar without further justification. First we show that context-free grammars, string-rewrite systems, and tree-adjoining grammars are all special cases of higher-dimensional rewriting systems called weak *n*-categories. Then we provide a "circuit-growing" grammar in terms of a weak *n*-categorical signature that simultaneously generates strings of grammatically acceptable text and its "deep structure" in terms of text circuits, from which we obtain the desired correspondence between text circuits and text. We close with demonstrations of how the syntactic theory of text circuits may be modified and expanded, and a brief note on how text circuits relate to Montague's "Universal Grammar".

1.1 A generative grammar for text circuits

1.1.1 A circuit-growing grammar

Definition 1.1.1 (Lexicon)**.** We define a limited lexicon $\mathcal{L}$ to be a tuple of disjoint finite sets $(\mathbf{N}, \mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_{\mathtt{S}}, \mathbf{A}_{\mathtt{N}}, \mathbf{A}_{\mathtt{V}}, \mathbf{C})$

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*
- $\mathbf{V}_{\mathtt{S}}$ is a set of *sentential-complement verbs*
- $\mathbf{A}_{\mathtt{N}}$ is a set of *adjectives*
- $\mathbf{A}_{\mathtt{V}}$ is a set of *adverbs*
- $\mathbf{C}$ is a set of *conjunctions*

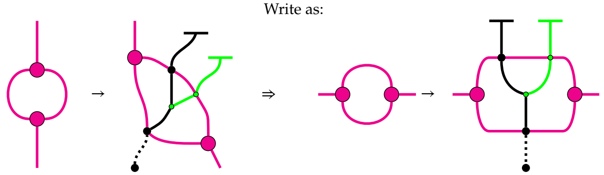


Figure 1.1: **How to read the diagrams in this section:** we will be making heavy use of pink and purple bubbles as frames to construct circuits. We will depict the bubbles horizontally, as we are permitted to by compact closure, or by reading diagrams with slightly skewed axes.

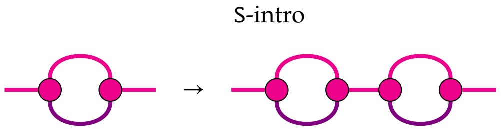


Figure 1.2: Every derivation starts with a single blank sentence bubble, to which we may append more blank sentences.

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 1.1.1, 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 1.3. 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.

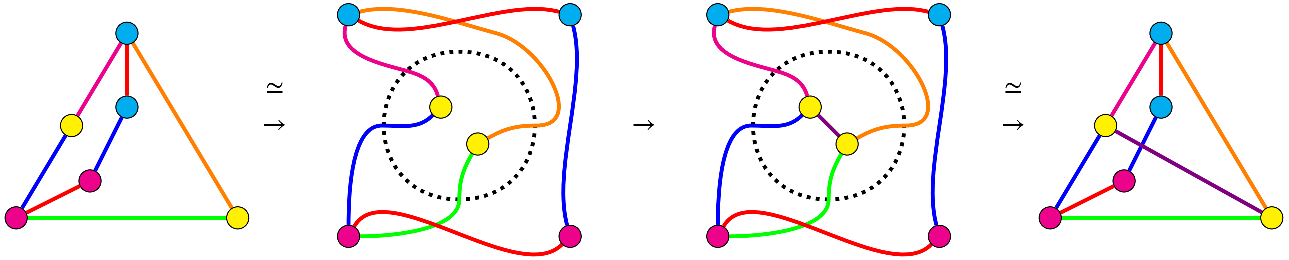


Figure 1.3: In this toy example, obtaining the same rewrite that connects the two yellow nodes with a purple wire using only graph-theoretically-local rewrites could potentially require an infinite family of rules for all possible configurations of pink and cyan nodes that separate the yellow, or would 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 ≃ notation.

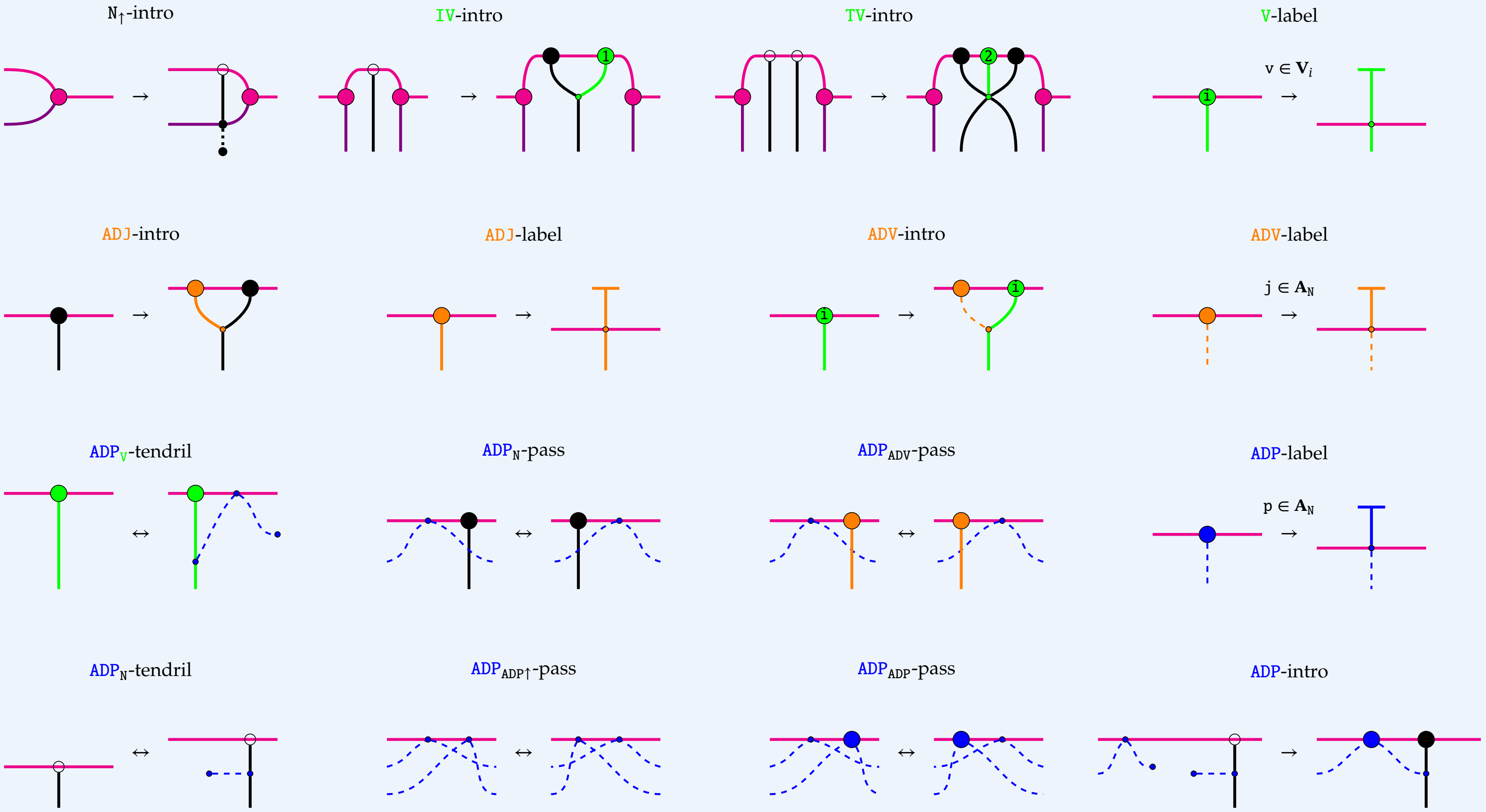
The broad plan: We'll first display the rules and demonstrate their usage, then we'll prove the text circuit theorem by relating our rules to text.

The rules: We start with simple sentences that only contain a single intransitive or transitive verb, which correspond to gates and typed-boxes. Then we consider more general sentences with nesting sentential structure, which correspond to untyped-boxes. Then we introduce coreferential structure on nouns, which corresponds to symmetric monoidal composition of text circuits.

The theorem: We characterise the expressive capacity of our rules for simple and complex sentences in terms of a context-sensitive grammar that corresponds to the surface structure of the derivations, which tells us that the generated text is sensible. Then we provide a mathematical characterisation of coreferential structure and a completeness result of our rules with respect to processive connectivity, which tells us that all circuit connectivity patterns are achievable. Then we (re)state and prove the text circuit theorem: that the fragment of language generated by the grammar surjects onto text circuits.

Aftermath: Finally, we examine how we may model extensions to the expressive capacity of text circuits by introduction of new rewrite rules.

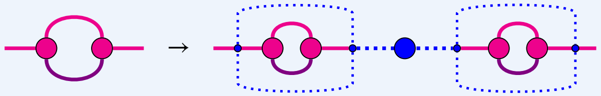
Rules 1.1.2 (Simple sentences). Simple sentences are sentences that only contain a single intransitive or transitive verb. Simple sentences will contain at least one noun, and may optionally contain adjectives, adverbs, and adpositions. The rules for generating simple sentences are as follows:


$\texttt{N}_{\uparrow}$-intro
IV-intro
TV-intro
V-label
$\mathtt{v} \in \mathbf{V}_i$
1
2
ADJ-intro
ADJ-label
ADV-intro
ADV-label
$\mathtt{j} \in \mathbf{A}_{\texttt{N}}$
$\texttt{ADP}_{\texttt{V}}$-tendril
$\texttt{ADP}_{\texttt{N}}$-pass
$\texttt{ADP}_{\texttt{ADV}}$-pass
ADP-label
$\mathtt{p} \in \mathbf{A}_{\texttt{N}}$
$\texttt{ADP}_{\texttt{N}}$-tendril
$\texttt{ADP}_{\texttt{ADP}\uparrow}$-pass
$\texttt{ADP}_{\texttt{ADP}}$-pass
ADP-intro

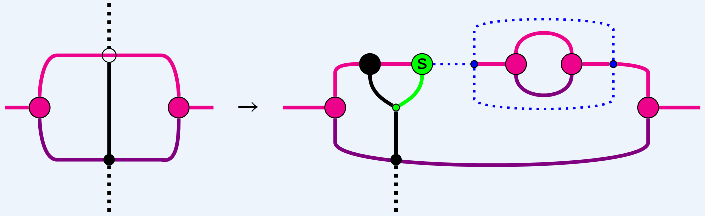
The $\texttt{N}_{\uparrow}$-intro rule introduces new unsaturated nouns. The IV-intro and TV-intro rules apply when there are precisely one or two unsaturated nouns in the sentence respectively, saturating their respective nouns. Adjectives may be introduced immediately preceding saturated nouns. Adverbs may be introduced immediately preceding verbs. To capture context-sensitive placement of adposition introductions, the $\texttt{ADP}_{\texttt{V}}$-tendril rule allows an unsaturated adposition to succeed a verb; a bulb may travel by homotopy to the right seeking an unsaturated noun. Conversely, the bidirectional $\texttt{ADP}_{\texttt{N}}$-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. 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 reversible.

Rules 1.1.3 (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.

CNJ-intro

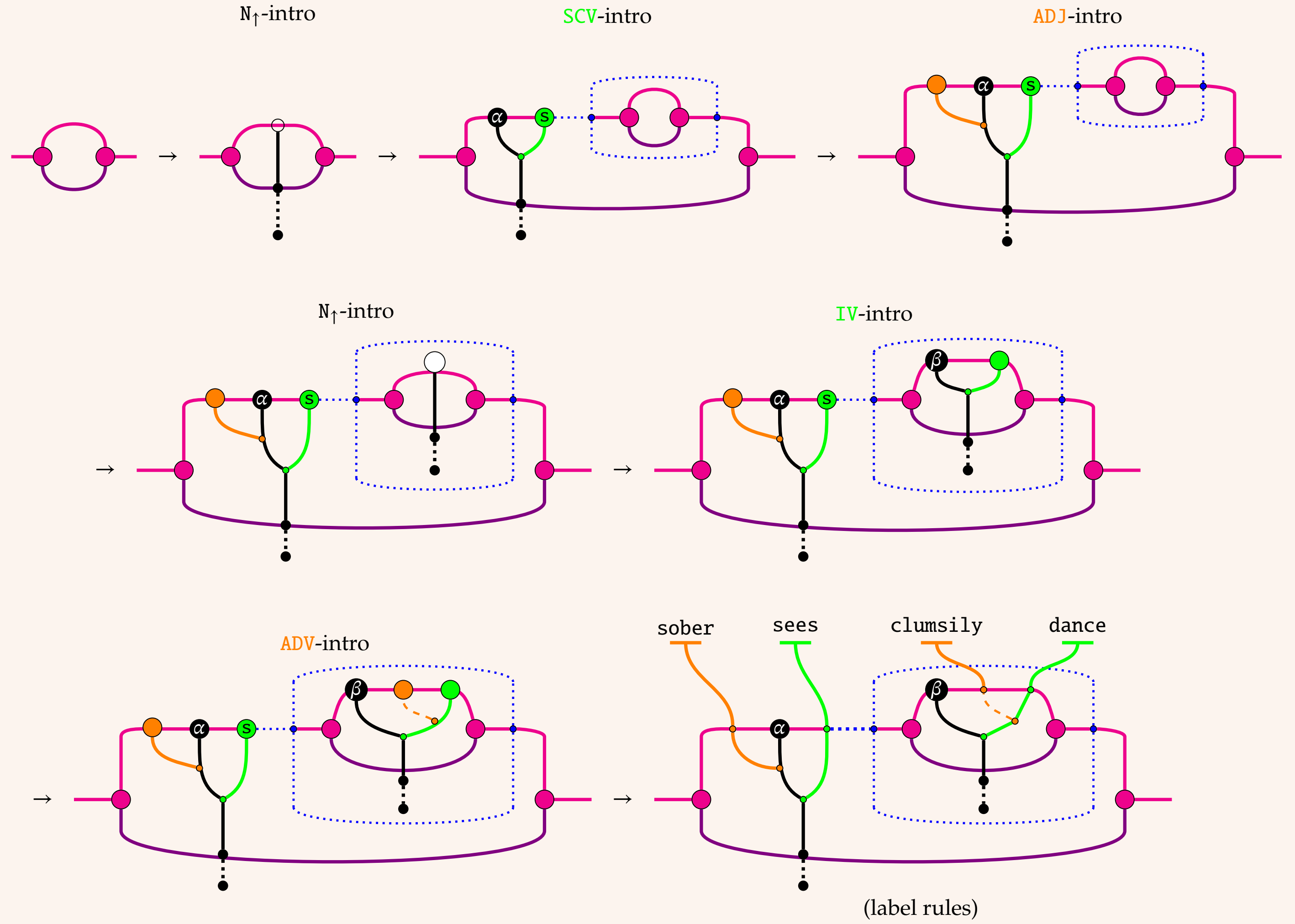


SCV-intro

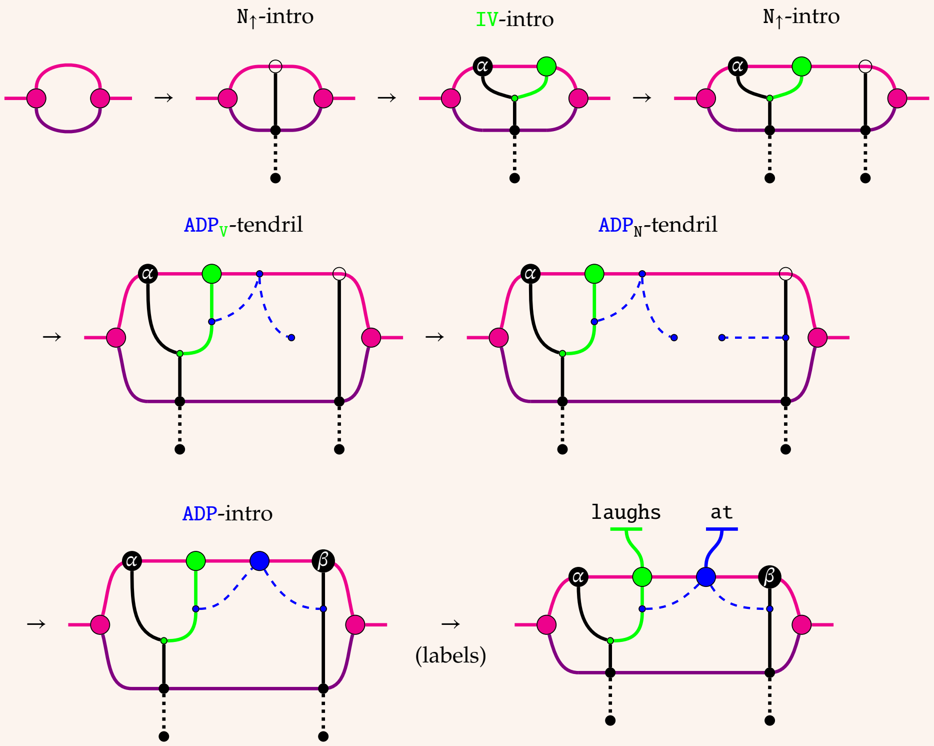


The dotted-blue wires do not contentfully interact with anything else, but this noninteraction disallows overgeneration cases where adpositional phrases might interject between `SCV` verbs and their sentential complement, e.g. `Alice sees at lunch Bob drink`. Later, they serve as visual indicators for the contents of untyped-boxes in text circuits.

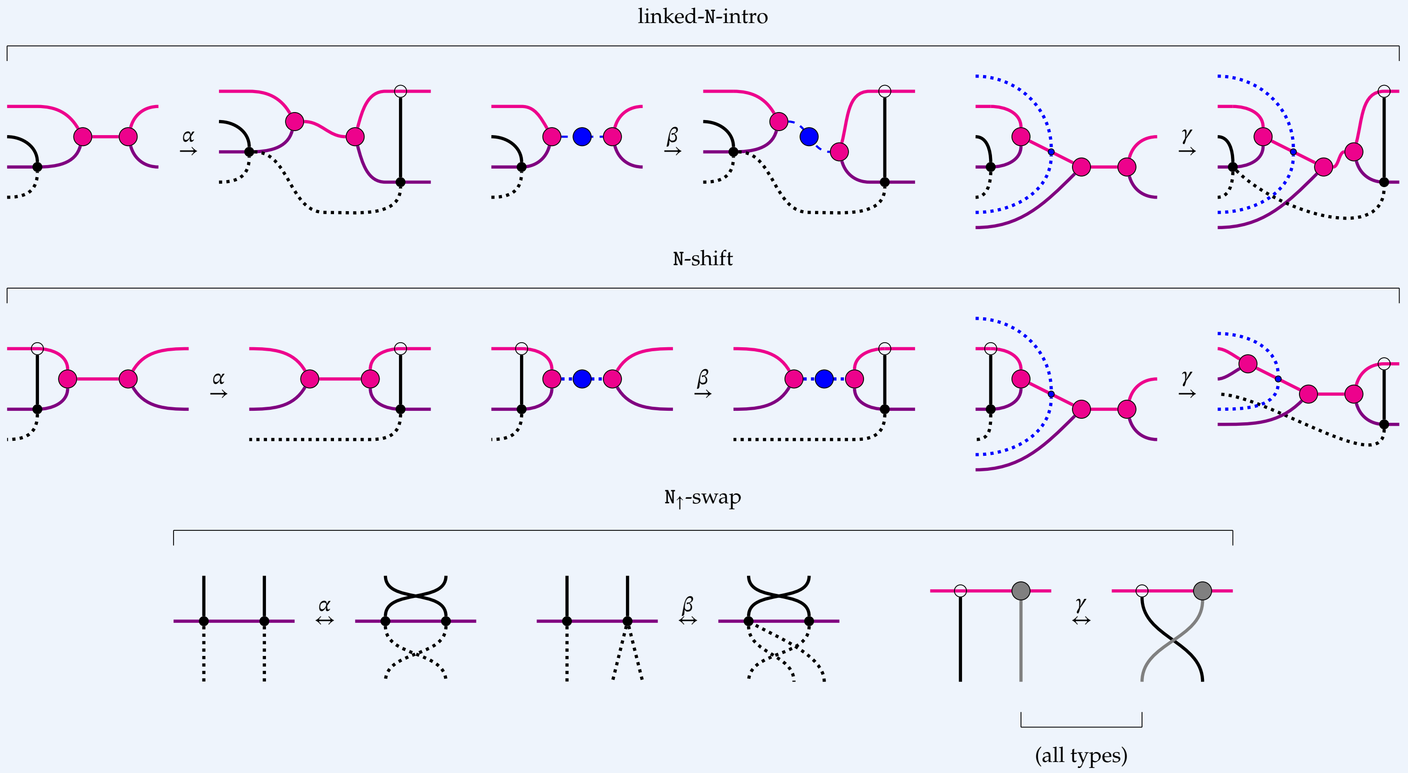
Example 1.1.4 (`sober` α `sees drunk` β `clumsily dance.`). Now we can see our rewrites in action for sentences. As a matter of convention – reflected in how the various pass- rules do not interact with labels – we assume that labelling occurs after all of the words are saturated. We have still not introduced rules for labelling nouns: we delay their consideration until we have settled coreferential structure. For now they are labelled informally with greeks.



Example 1.1.5 (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.

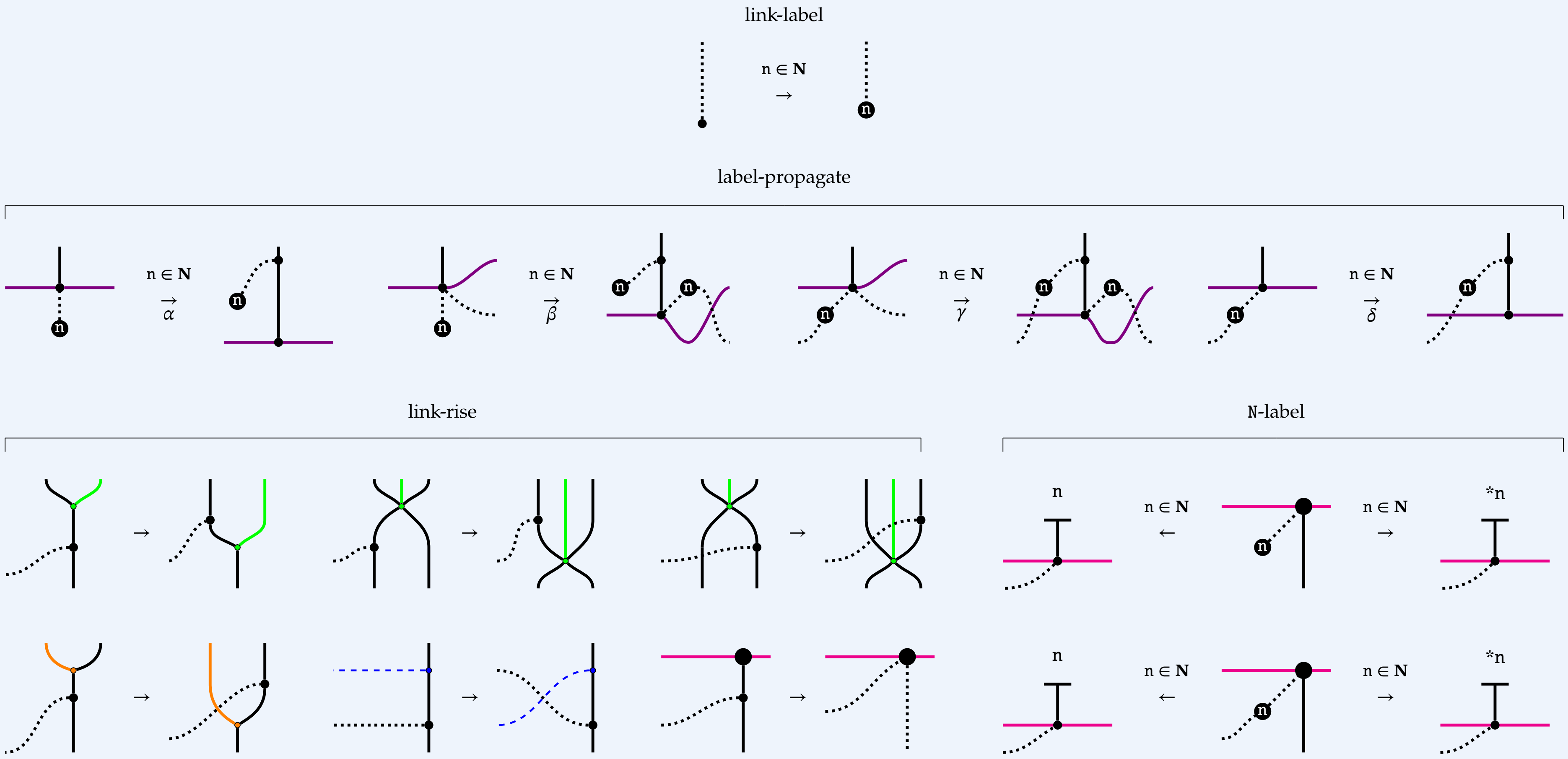


Rules 1.1.6 (Coreferential structure and noun labels).



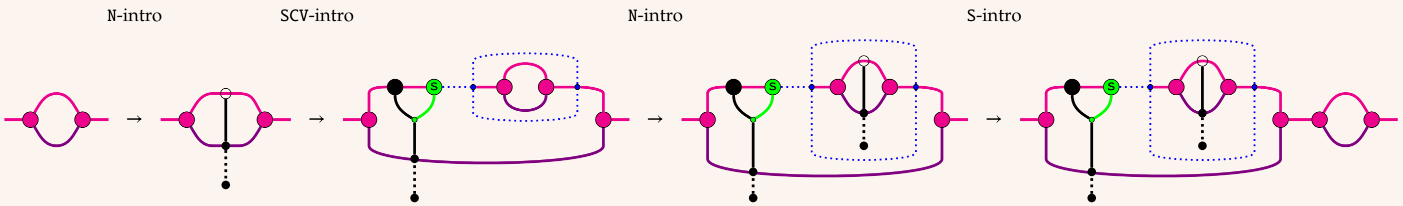
The linked-N-intro rules introduce a new unsaturated noun in the next sentence that coreferences the noun in the previous sentence that generated it, with variants for each pair of sentences involved. We depict three: simple to simple, between CNJ-related sentences, and from either a CNJ or SCV to a simple sentence.
N-shift rules allow any unsaturated noun to move into the next sentence, again with variants for different pairs of sentences. 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.

Rules 1.1.7 (Labelling nouns)**.** 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:

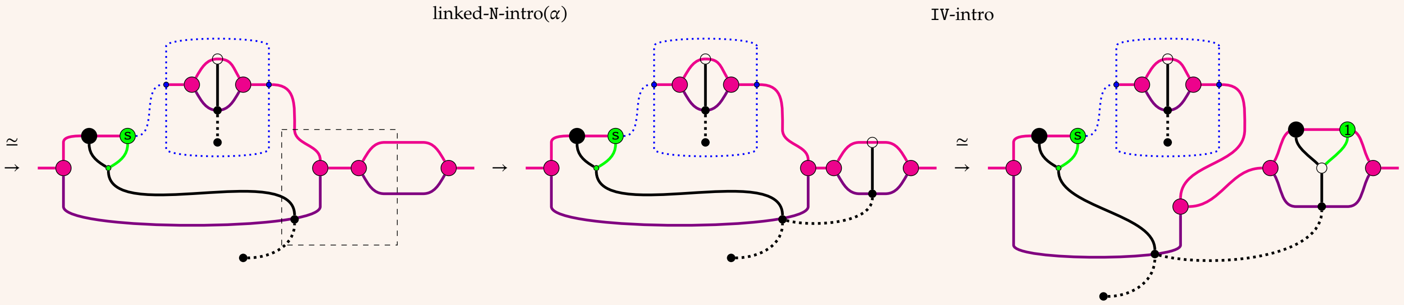


The $n \in \mathbf{N}$ notation indicates a family of rewrites (and generators) for each noun in the lexicon. Link-label assigns a noun to a diagrammatically linked collection of coreferent nouns, and link-propagation is a case analysis that copies a link label and distributes is across coreferent nouns. 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 n or a pronoun appropriate for the noun, notated *n

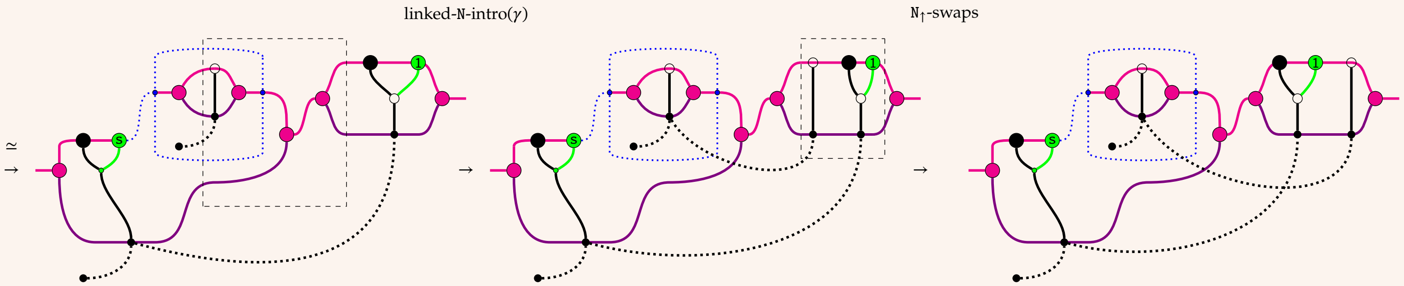
Example 1.1.8 (sober Alice sees Bob clumsily dance. She laughs at him.). We start the derivation by setting up the sentence structure using S- and SCV-intro rules, and two instances of N-intro, one for Alice, and one for Bob. Observe how the N-intro for Bob occurs within the subsentence scoped over by the SCV-rule.



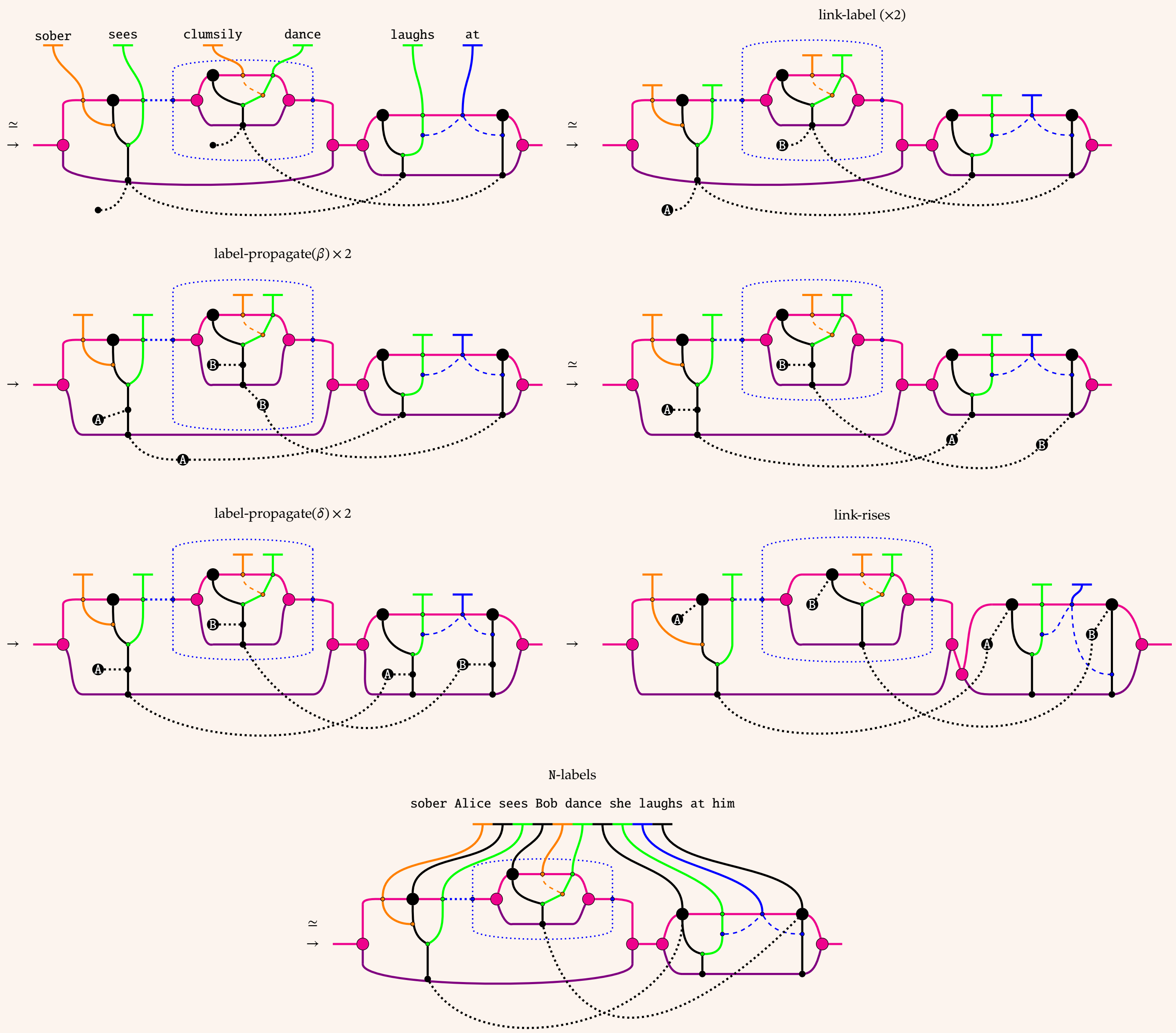
By homotopy, we can rearrange the previous diagram to obtain the source of the linked-N-intro rewrite in the dashed-box visual aid. Observe how we drag in the root of what is to be Alice's wire. Then we use the IV-intro in the second sentence, which sets up the surface structure she laughs, and the deep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



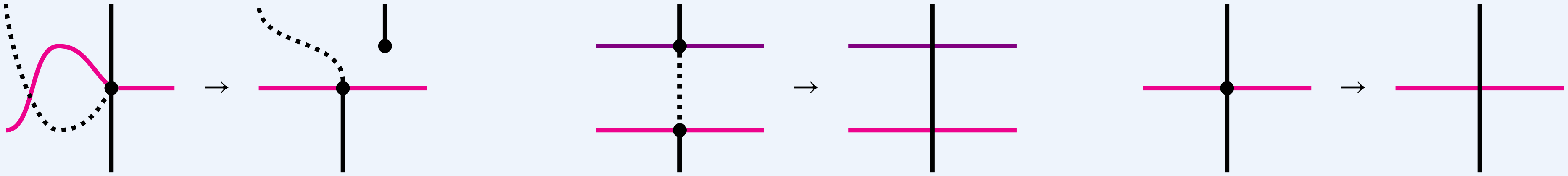
By homotopy again, we can do the same for Bob, this time setting up for the γ variant of linked-N-intro which handles the case when the spawning noun is within the scope of an SCV. Then by applying a series of N$_{\uparrow}$-swaps, the unsaturated noun is placed to the right of the intransitive verb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



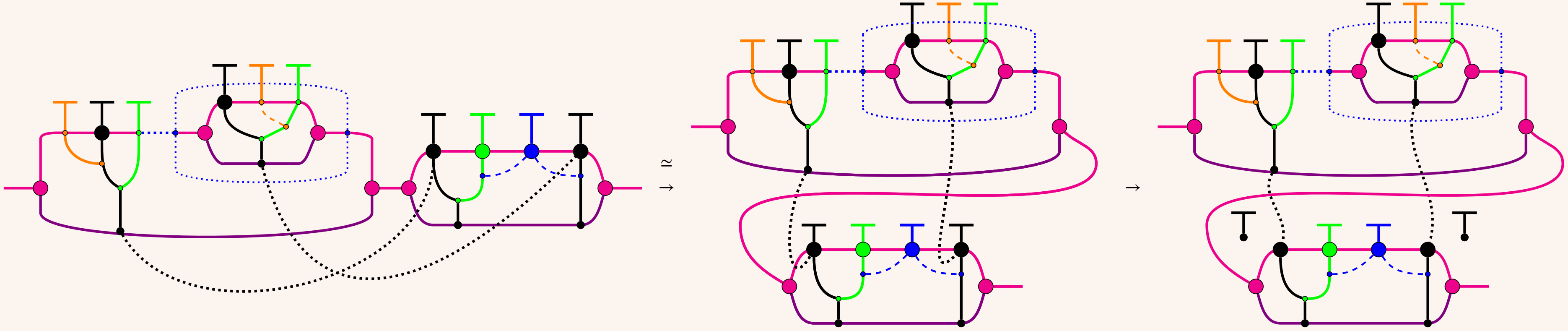
We've already done the surface derivation for the two sentences separately in Examples 1.1.4 and 1.1.5; since neither of those derivations touch the roots of noun-wires, we can emulate those derivations and skip ahead.


sober
sees
clumsily
dance
laughs
at
link-label (×2)
label-propagate(β) × 2
label-propagate(δ) × 2
link-rises
N-labels
sober Alice sees Bob dance she laughs at him
A
B

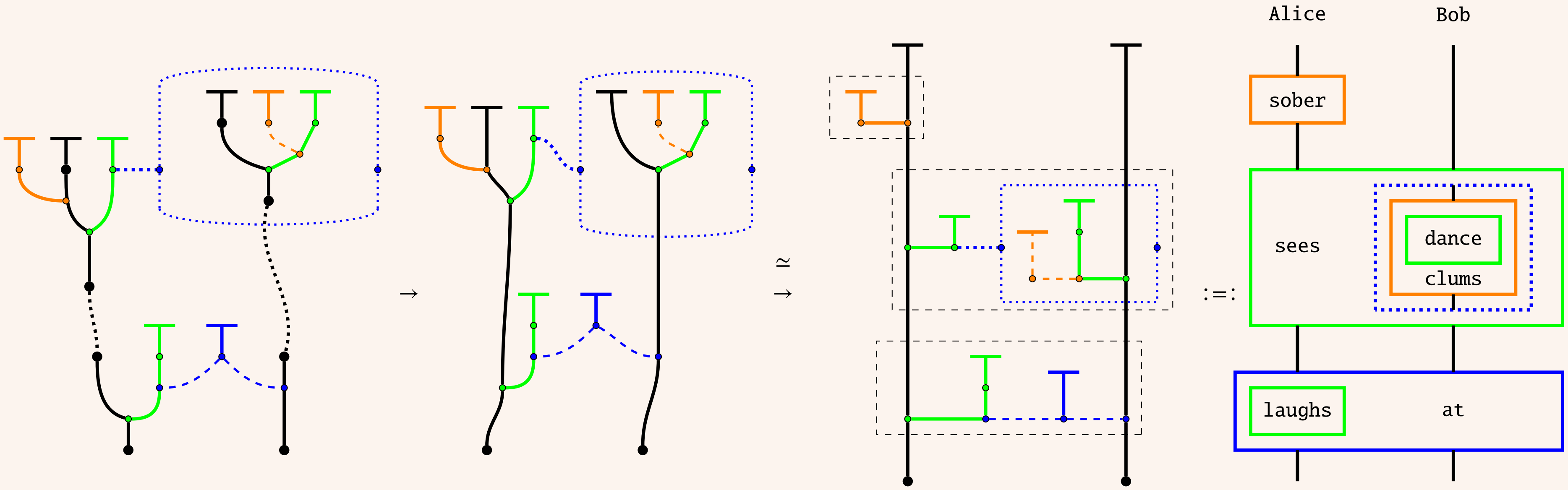
Rules 1.1.9 (Text to circuit)**.** We turn finished text diagrams into text circuits by operating *in situ*, with extra rules outside the grammatical system that handle connecting noun wires.



Example 1.1.10 (Text to circuit in action). In the first step below, by Lemmas 1.1.17 and 1.1.18, 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 1.1.9 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 to. In the fourth step, we apply the second and third rewrites of Construction 1.1.9 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*.



1.1.2 Text circuit theorem

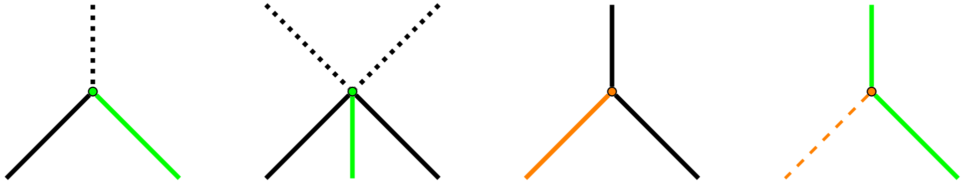


Figure 1.4:

Construction 1.1.11 (CSG for simple sentences)**.**

We may gauge the expressivity of simple sentences with the following context-sensitive grammar.

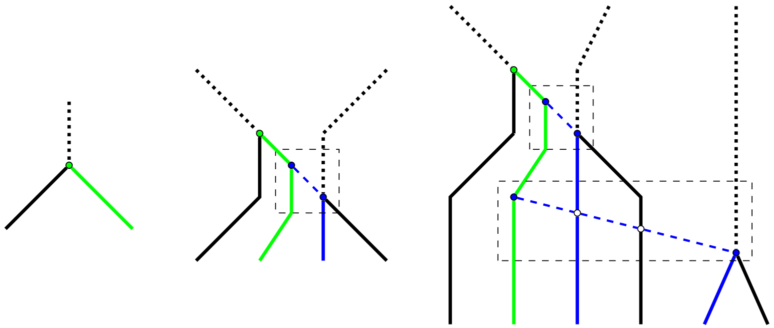


Figure 1.5: Adpositions require several helper-generators; we depict for example the beginning of the sequence of derivations that result from appending adpositions to an intransitive verb (the generators are implicit in the derivations):

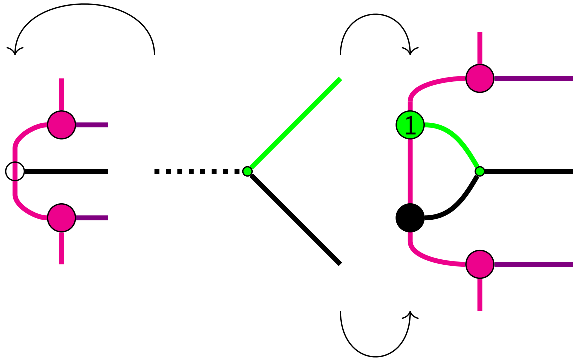


Figure 1.6:

Proposition 1.1.12. 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. Reading top-down, the dashed-blue helper lines for adpositions indicate ADP-pass rules. The correspondence is visually evident, but may be expressed as a separate signature. The correspondence IV-intro is depicted, to be read from left-to-right. □

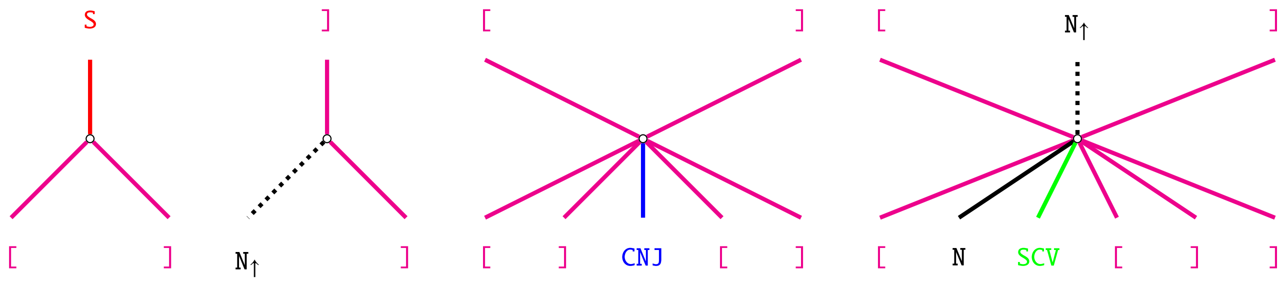


Figure 1.7:

Proposition 1.1.13. Up to labels, Rules 1.1.2 and 1.1.3 for simple and complex sentence yield the same sentences as the combined CSG of Construction 1.4 with the depicted additional rules.

Proof. Same correspondence as Proposition 1.6, ignoring the dotted-blue guards. □

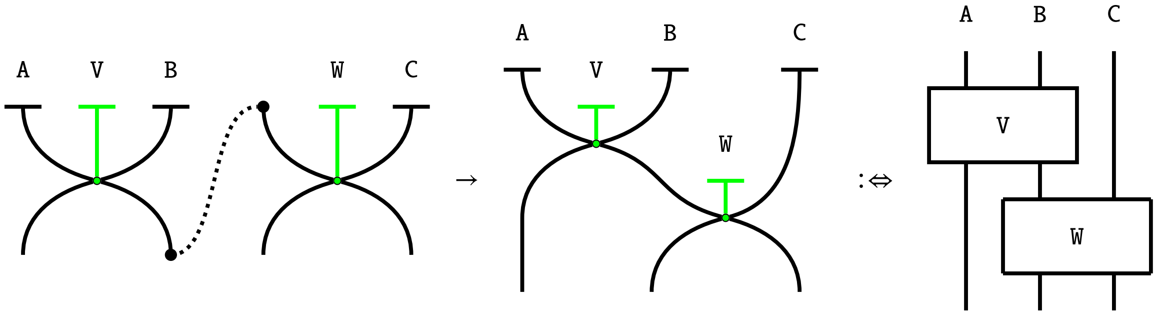


Figure 1.8: We choose the convention of connecting from left-to-right and from bottom-to-top, so that we might read circuits as we would text: the components corresponding to words will be arranged left-to-right and top-to-bottom. Connecting nouns across distinct sentences presents no issue, but a complication arises when connecting nouns within the same sentence as with reflexive pronouns e.g. `Alice likes herself`.

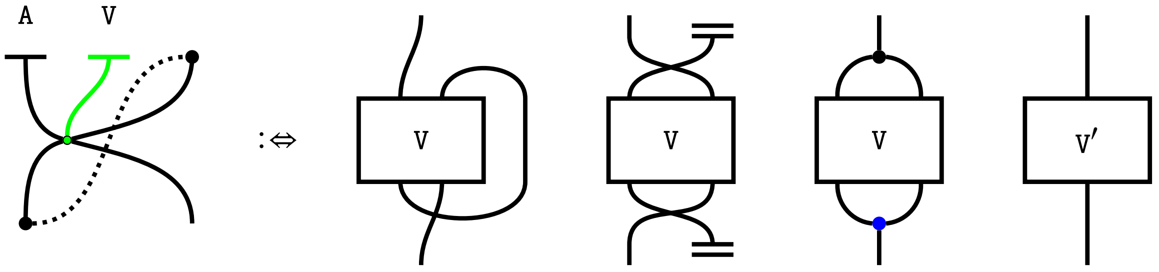


Figure 1.9: Reflexive coreference would violate of the processivity condition of string diagrams for symmetric monoidal categories. Not all symmetric monoidal categories possess the appropriate structure to interpret such reflexive pronouns, but there exist interpretative options. From left to right 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 provide a purely syntactic treatment in CITE ; for now we treat them as if they were just verbs of lower arity.

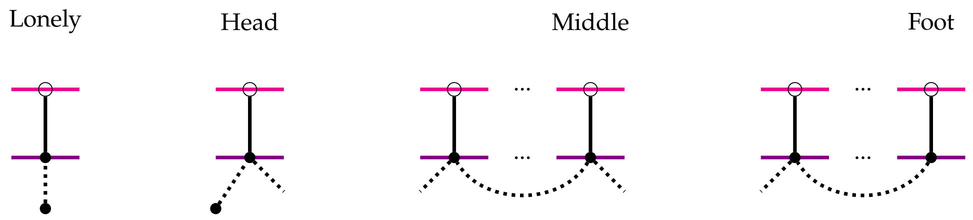


Figure 1.10:

Terminology 1.1.14 (Kinds of nouns with respect to coreference)**.** 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.

Definition 1.1.15 (Finished text diagram)**.** The circuit-growing grammar produces *text diagrams*. We call a text diagram *finished* if all surface nodes are labelled.

Proposition 1.1.16. 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 1.6 and 1.7. 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 1.1.17. The unsaturated noun kinds listed in Figure 1.10 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. □

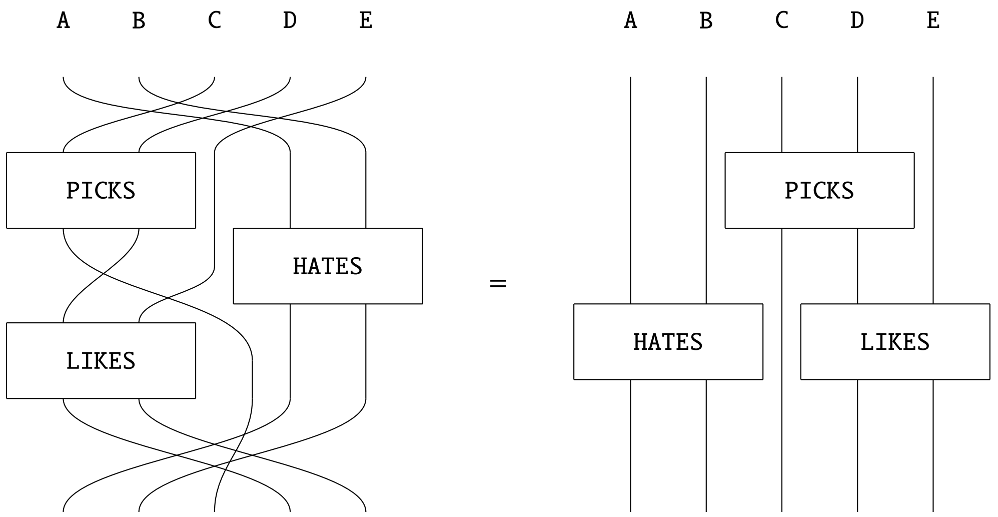
Lemma 1.1.18. No nouns within the same sentence are coreferentially linked.

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. □

Proposition 1.1.19 (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 1.6 and 1.7. Lemmas 1.1.17 and 1.1.18 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 1.1.9, T can be 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 1.1.19 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.



Remark 1.1.20. There are some oddities about our conventions that will make sense later when we consider semantics. For example, Convention 1.11 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 1.14 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 1.1.22 between typed and "untyped" higher-order processes will be given a suitable semantic interpretation in Section lassos .

Figure 1.11:

Convention 1.1.21 (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.

Convention 1.1.22 (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 1.1.24 (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.

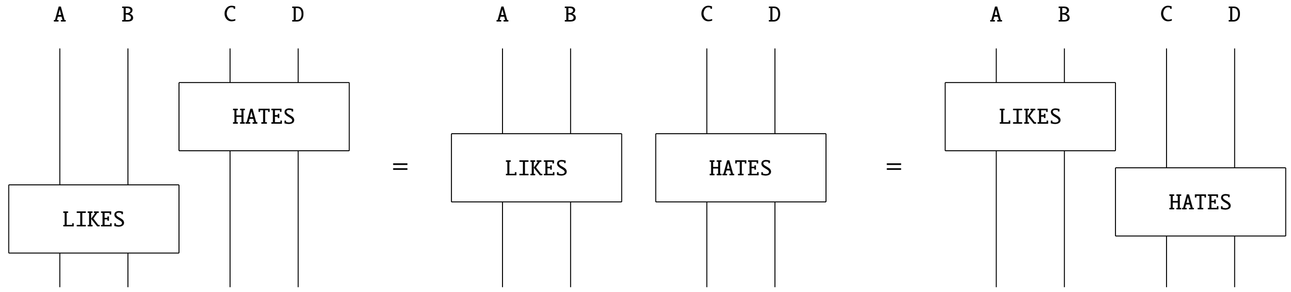


Figure 1.12:

Convention 1.1.23 (Sliding)**.** Since only gate-connectivity matters, we consider circuits the same if all that differs is the horizontal positioning of gates composed in parallel.

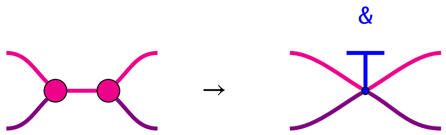


Figure 1.13:

Convention 1.1.25 (Contentless conjunctions)**.** Conventions 1.12 and 1.1.24 require something else to allow them to work at the same time: something to distinguish when the gates are parallel. So we ask for "contentless" conjunctions, such as `and`, or `while`. In terms of text diagrams, we want rewrites that introduces such contentless conjunctions and witnesses their associativity, like this:

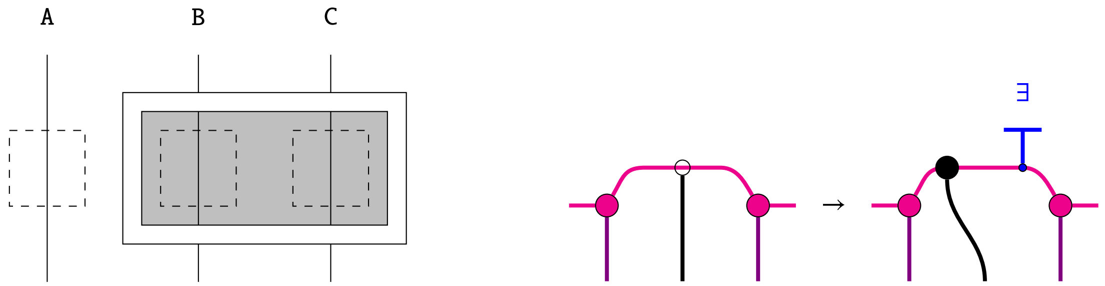
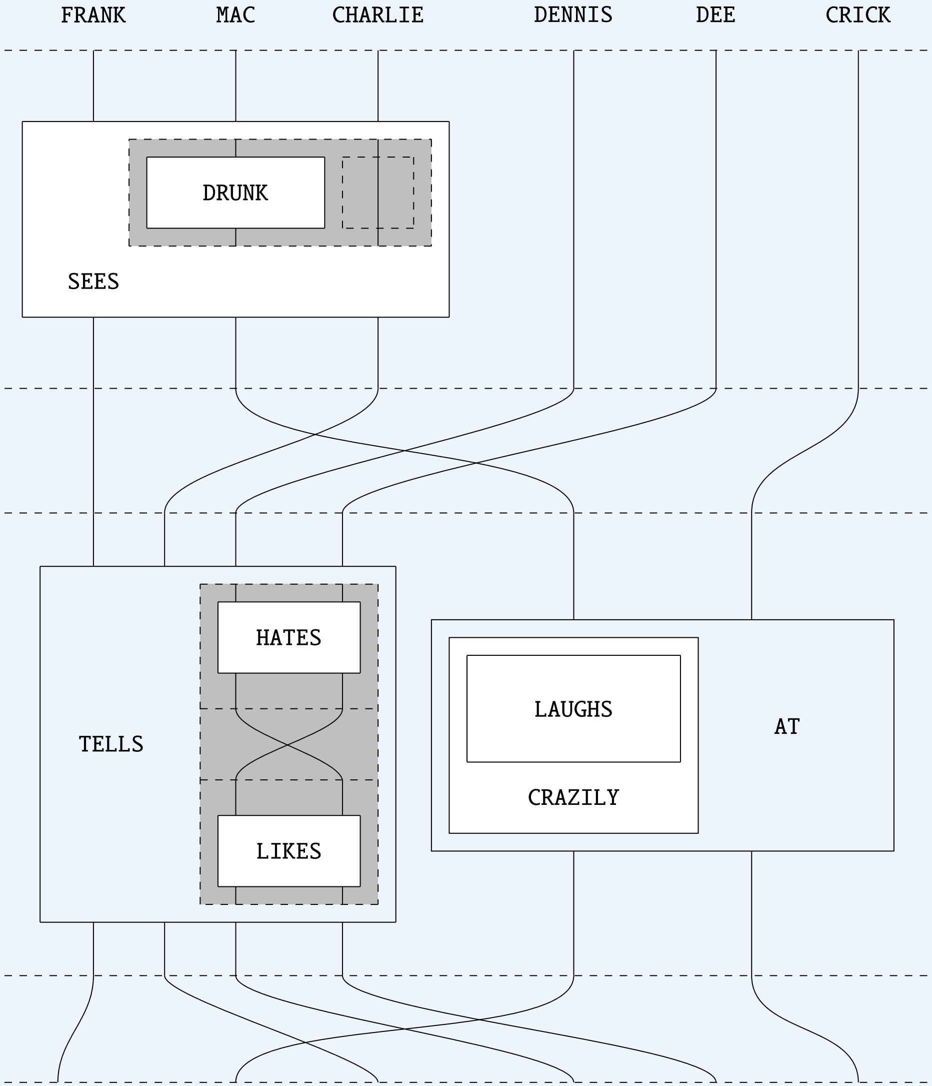


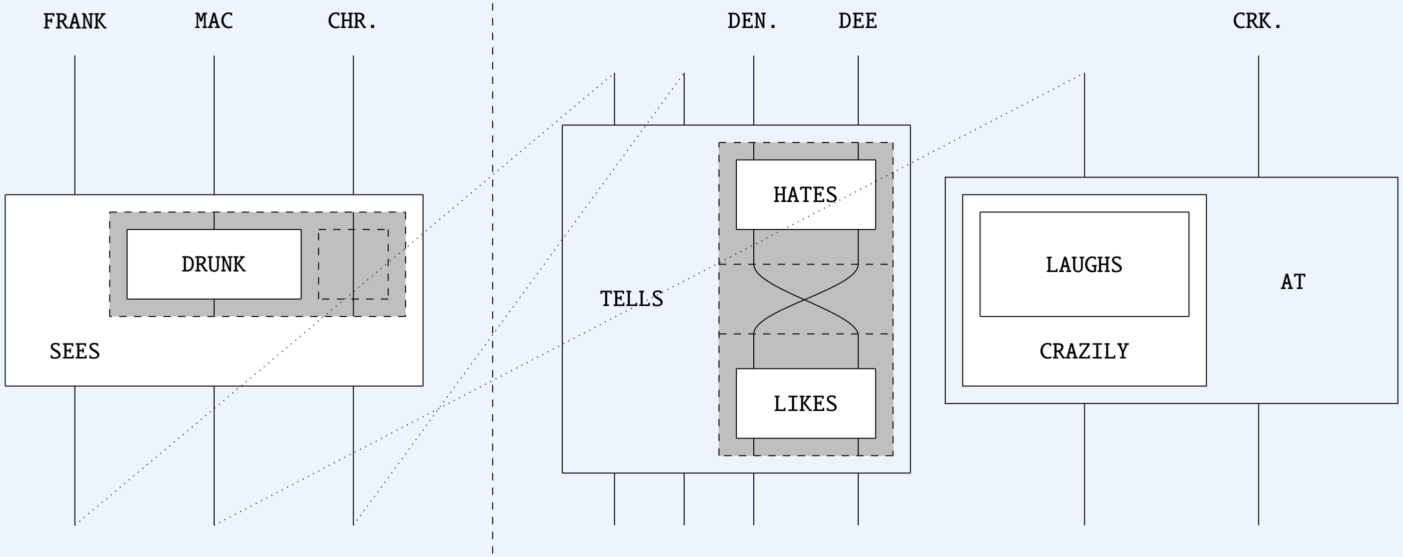
Figure 1.14:

Convention 1.1.26 (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 (left). 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`. So to deal with lonely wires in terms of text diagrams, we want a rewrite that introduces such contentless verbs (right).

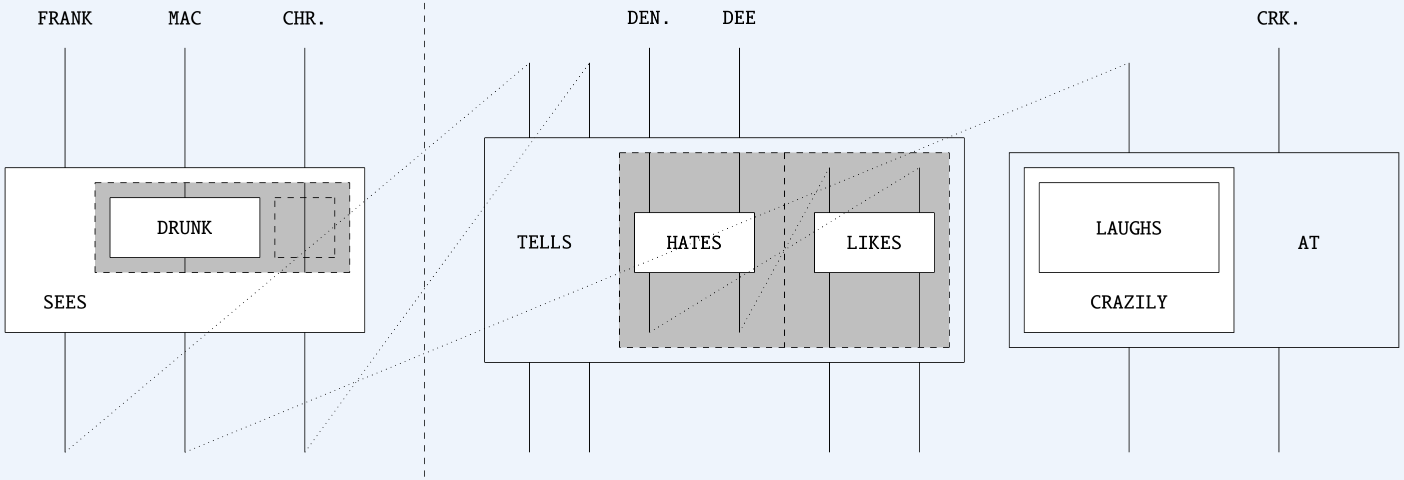
Construction 1.1.27 (Circuit to text)**.** In the presence of additional rewrites from Conventions 1.13 and 1.14, every text circuit is obtainable from some text diagram, up to Conventions 1.11 and 1.12. Starting with a circuit, we may use Convention 1.11 to arrange the circuit into alternating slices of twisting wires and (possibly tensored) circuits, and this arrangement recurses within boxes. Slices with multiple tensored gates will be treated using Convention 1.13. By convention 1.14, 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 1.1.22.



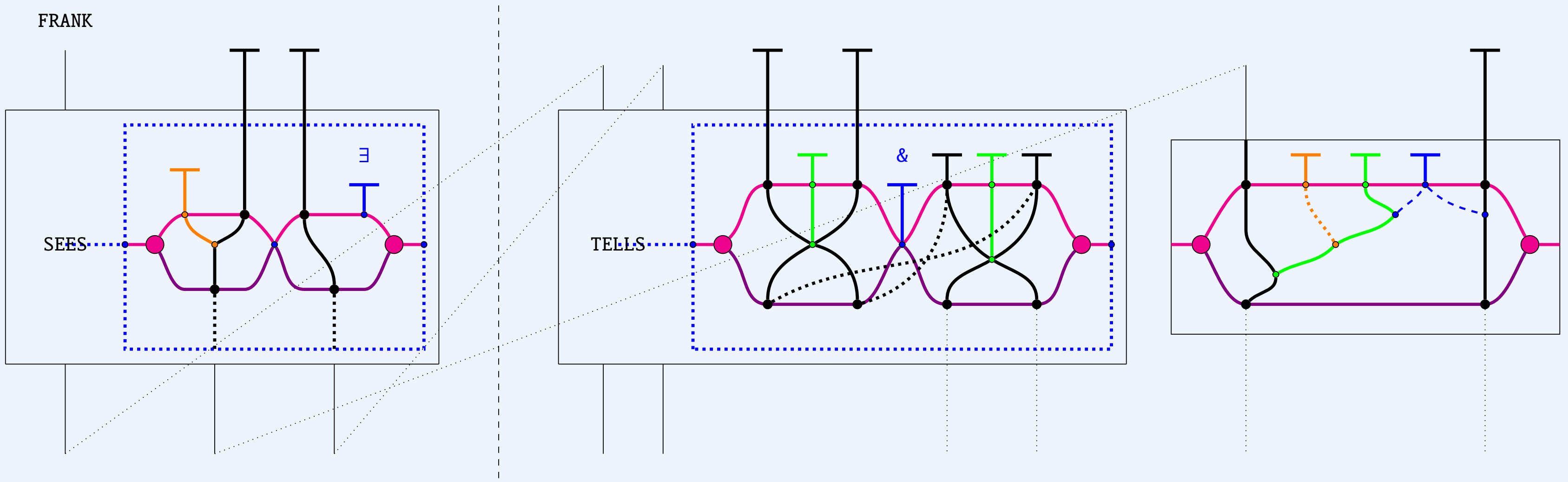
We then linearise the slices, representing top-to-bottom composition as left-to-right. Twist layers are eliminated, replaced instead by dotted connections indicating processive connectivity. The dashed vertical line distinguishes slices. This step of the procedure always behaves well, guaranteed by Proposition 1.1.17. Noun wires that do not participate in earlier slices can be shifted right until the slice they are introduced.



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.



By Proposition 1.7, gates are equivalent to sentences up to notation, so we swap notations *in situ*. Conventions 1.13 and 1.14 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.

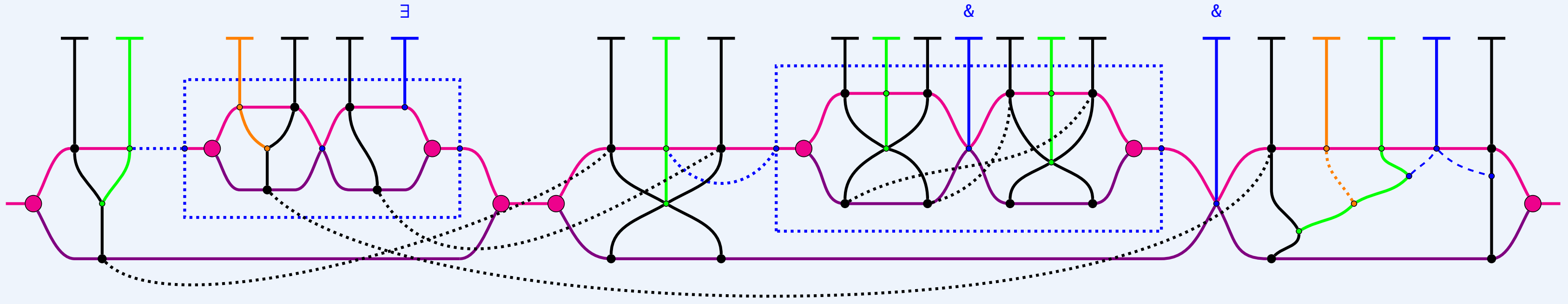


Recursing notation swaps outwards and connecting left-to-right slices as sentence-bubbles connect yields a text circuit, up to the inclusion of rewrites from Conventions 1.13 and 1.14: applying the reverse of those rewrites and the reverse of text-diagram rewrites yields a valid text-diagram derivation, by Propositions 1.7 and 1.1.17. We haven't formally included transitive verbs with sentential complement in our vocabulary, but it should be obvious at this point how they function with our existing machinery.

Frank sees [drunk Mac (&) Charlie (∃)].

Frank tells Charlie [Dennis hates Dee (&) Dee likes Dennis]

(&) Mac crazily laughs at Cricket.



Theorem 1.1.28 (Text Circuit Theorem)**.** Text generated by the circuit-growing grammar is sensible and surjects onto text circuits. Hence:

Text circuits are a generative grammar for text

Proof. Sensibility at the sentential level is established by Propositions 1.6 and 1.7. Proposition 1.1.19 establishes a map from text to circuits, and Construction 1.1.27 witnesses its surjectivity. The conventions required for the construction are accompanied by justifications of sensibility. □

1.2 *Text circuits: details and development*

This section covers some practical developments, references for technical details of text circuits, and sketches of how to play with them. 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 CITE . 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, detailed in the margin. 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.

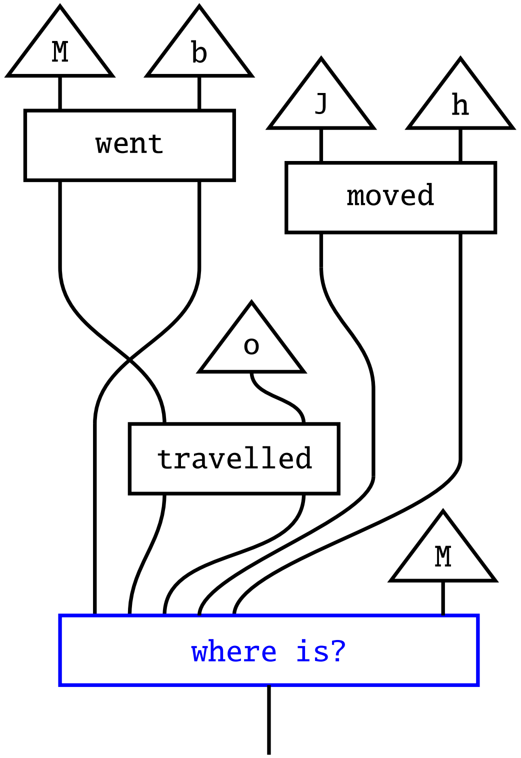
In practice, the process of obtaining transparently computable text goes through two phases. First, one has to obtain text circuits from text, which is conceptually simple: typelogical parsers for sentences can be modified to produce circuit-components rather than trees, and a separate pronomial resolution 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; an example of this interpretation of families of processes is the use of an aggregation monoid in graph neural network CITE . 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 that indicates related families of morphisms 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 theo-

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 in the form of a process-theoretic equation to be satisfied: the depicted composite process ought to be equal to the `office` state:



rem. 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 computing with text, where facets are open to interpretation and modification.

1.3 How to play

Text diagrams of the sort we have been growing with the circuit-growing grammar have rich structure for modelling syntactic structure. In tandem with an implementation in neural nets, there are many things that can be achieved. Here I want to share some possibilities and general principles.

Functional words. No separate "information structure" is required; all is transparent and manipulable. For instance, in the case of relative pronouns in Example 1.3.2, ansatz wires may be freely introduced and manipulated to achieve the desired correspondence between deep and surface structure.

Grammar equations. A principle for the extension of text diagrams to larger fragments of text is to devise *grammar equations* CITE that express novel textual elements in terms of known text diagrams, for instance, that the use of passive voice or copulas amounts to swaps in the linear surface order, as in Examples 1.3.3 and 1.3.4. Syntactic rules may even introduce semantic components, as in for instance the treatment of the possessive pronoun in Example 1.3.5. In sum, intuitions about the systematicity of language are directly translatable into rewrite rules that manipulate deep structure as one sees fit.

Grammatical typing is nesting. A heuristic for the extension of text diagrams to novel grammatical categories is the absorption of type-theoretical compositional constraints at the level of sentences into nesting of boxes, as demonstrated in Examples 1.3.6 and 1.3.7.

Syncategorematicity. A useful heuristic for the application of text diagrams is to treat individual text circuits as analogous to propositional contents, and certain logical or temporal connectives as structural operations upon circuits – rewrites – that must be applied in order to obtain a purely propositional format. In other words, logical or structural words are to be treated as circuit-manipulation instructions to be executed in order to obtain a circuit, in the same way that $1 + 1$ is only an integer expression once addition has been evaluated. See Examples 1.3.8 and 1.3.9 for a demonstration.

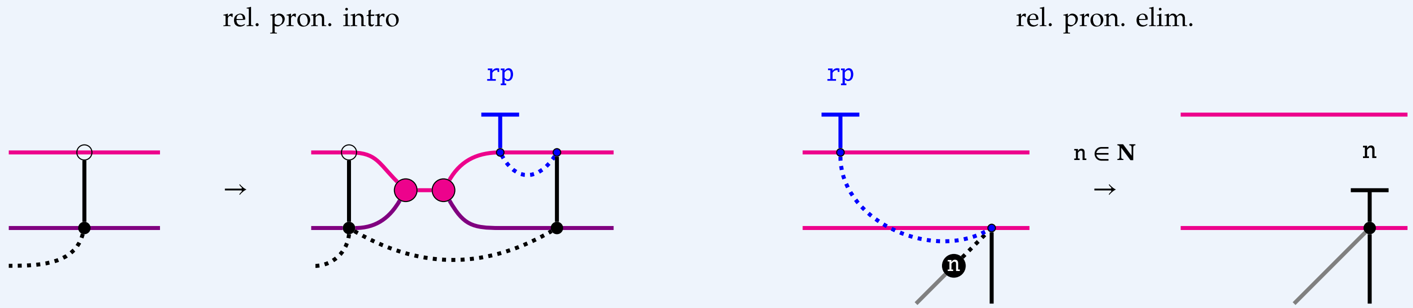
Syntax in context. Extending the reach of text circuits to determiners, quantifiers, and conditionals appears to require a contextual process theory in which to evaluate and enforce constraints upon the purely syntactic content of text circuits. The broad strategy sketched here rests upon three tactics. First, as in the neural approach to bAbi, word-gates are considered to be paired with measurement-processes that return an analog of truth values, the latter of which may be generic tests for adjectives as static predicates or verbs as dynamic predicates. The pairing of gates with measurements follows the philosophy of update structures CITE . The truth-measurements allow conditionals to be expressed as either circuit-rewrites or constraints on truth-measurements, the latter which are in turn interpretable as loss-functions in the process of training gates. Second, we model context as the rest of the text circuit, which is a modifiably finite model. Third, we suppose we have a way to record and relate alternative circuits. These tactics appear sufficient for a first pass. Determiners may be considered to be context-sensitive connectivity. Universal quantifiers may be analysed in particular finitary contexts as conditionals and constraints on truth-conditional measurements. Existential quantifiers evaluated in the finitary case yield alternative circuits. See Examples 1.3.11, 1.3.12, 1.3.13, 1.3.15, and 1.3.15 for demonstrations.

Text is composition. Apart from nesting, the other form of composition available for text circuits is the connectivity of wires. We have presented a simplified theory of discourse where the only discourse referents are nouns, but this is not an inherent limitation of text diagrams, where grammatical data of all kinds are freely presentable and composable. In this presentation, I have stuck to string-diagrams in a compact-closed setting and their rewrites, and I have avoided the affordance of weak n-categories to specify *manifold* diagrams and their rewrites, where in addition to 1-D strings connecting 0-D boxes, there may be 2-D planes connecting 1-D strings, 3-D volumes connecting planes, string diagrams restricted to surfaces or volumes... All this is to say that if you can draw it, language can have whatever geometry you want. It just so happens that symmetric monoidal categories are the royal road for pedagogy and practicality (who knows how to interpret a manifold diagram as a computational process?), so it is best to stick to strings.

Objection!: Hold on, isn't this just transformational grammar? Haven't we moved on from that? In spirit, yes. The two major mathematical distinctions here are well-typing and many-input-many-output instead of treelike. The practical distinction is that this theory works with neural nets. Both approaches have the same theoretical problems: over- and undergeneration, no evidentiary basis for psychological realism, too rigid for functionalists, and so on. But recall that we differ in aims: our formalist approach is ultimately in service of approximating human language structure in machines for interpretability. How so? Solving language tasks such as bAbi via text circuits 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. What is meant by "conceptually-compliant" is a stronger variant of Firth's maxim: "the meaning of a word is how it interacts with other words". How do we justify that claim? The initial conception of bAbi was that the ability to answer questions about – for instance, the verb `to go` – in many different contexts amounted to having a consistent internal "world-model". But question-answering performance by itself is evidently insufficient for the degree of interpretability implied by conceptual-compliance, because the internal model is not forthcoming in transformer solutions. On the other hand, we *do* obtain the building blocks of compositional world-models by learning word gates in text circuits: each learnt word gate may be considered a well-grounded semantic primitive in the construction of novel text circuits, and the resulting circuits are modifiable world-models that are queriable using the (also learnt) measurement-gates. Why is that so? Because just as in Section REF we do not need to know how an update is implemented if it satisfies characteristic operational constraints imposed by process-theoretic equations, we don't need to know what's going on inside the gate `to go` so long as it satisfies the process-theoretic equations that `to go` ought to satisfy. What are these equations? Firth says that it is how `to go` behaves with respect to all other words in all contexts, which we approximate by translating individual bAbi tasks involving the word `to go`, via text circuits, into a representative sample of the process-theoretic equations that `to go` ought to satisfy. So the philosophical strength of the claim that `to go` and synonyms have been learnt-from-data in a way that coheres with human conceptions rests on three points: performance, Firth (or if you like, the Yoneda Lemma), and the breadth and variety exhibited in the bAbi dataset. The real test is practical demonstration, for which time will tell.

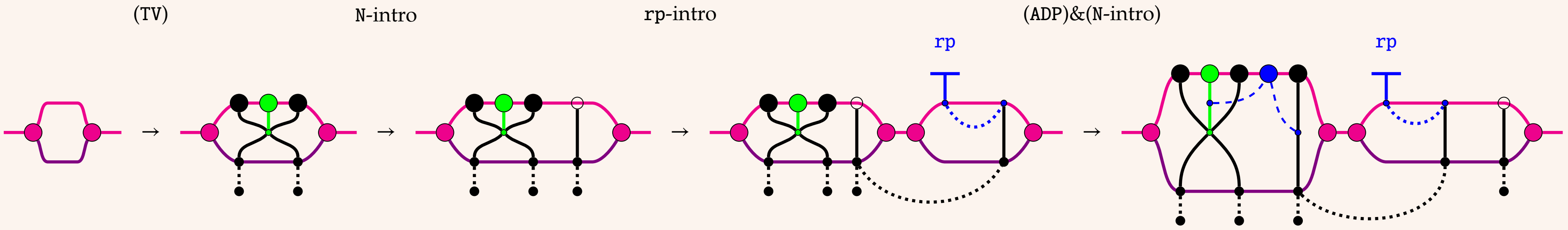
For the examples to follow, blue boxes denote rules already known, and the orange are either novel or worked examples.

Rules 1.3.1. A relative pronoun glues two sentences across a single noun. For example, what would be a text with two sentences `Alice teaches at school. School bores Bob.` can be composed by identifying `school`: `Alice teaches at school that bores Bob.` In this case, `that` is called a subject-relative pronoun, as it stands in for the subject argument in `bores`. In `Alices teaches at school that Bob attends.`, we have an object-relative pronoun, as `that` stands in for the object argument in `attends`. In English, relative pronouns occur immediately after the noun they copy, and are followed immediately by a sentence missing a noun. Reflexive pronouns may be dealt with in a similar fashion as we have with relative pronouns, and various semantic interpretation options have been covered already in Figure 1.9. We can extend our system to accommodate relative pronouns as follows. The introduction of a relative pronoun is considered to start a new sentence with a premade pronominal link. The relative pronoun connects to a new kind of surface noun, which for most intents and purposes behaves just like the nouns we have seen before, except for two caveats: it cannot be labelled (since the relative pronoun is already handling that), and it cannot leave the sentence it starts in by `N`-shift rules. We can eliminate relative pronouns by forcing the governed noun to be named, which is equivalent to dropping the relative pronoun and recovering distinct sentences.

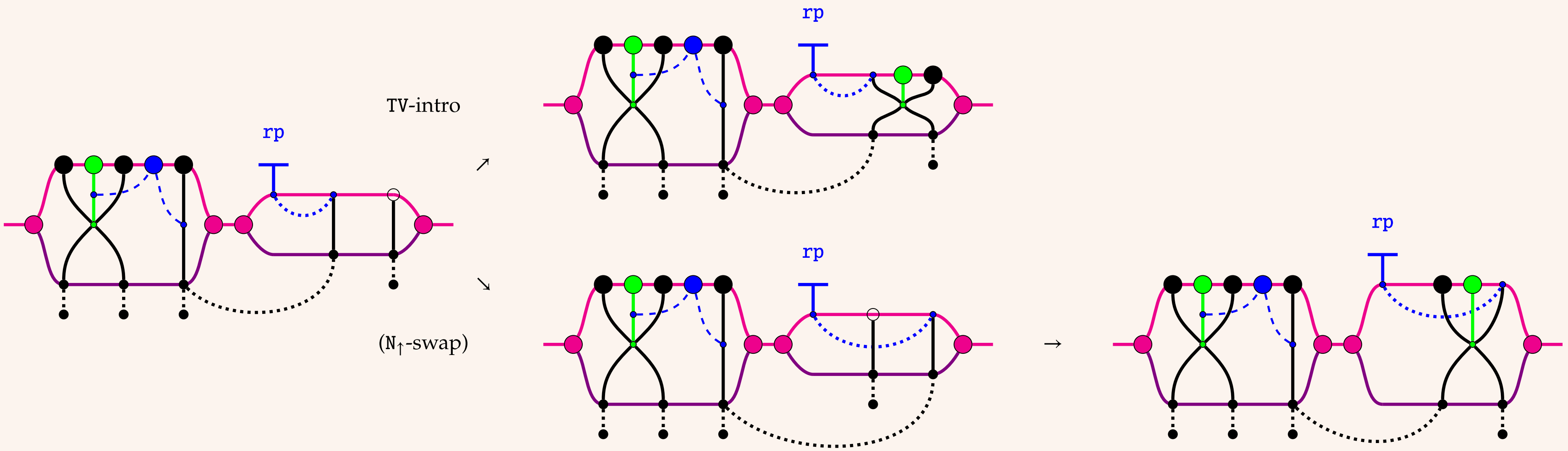

rel. pron. intro

rel. pron. elim.

Example 1.3.2 (Introducing relative pronouns). Here we demonstrate derivations of `Alice teaches at school that bores Bob` and `Alice teaches at school that Bob attends`. The initial steps in both cases are the same, setting up the `teaches` phrase structure and introducing a new unsaturated noun in the `Bob` phrase to work with the relative pronoun.



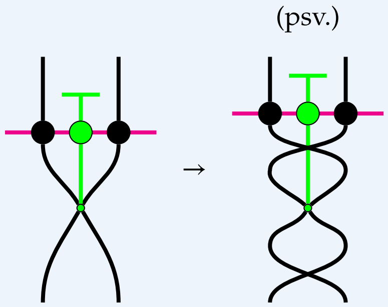
Now have a branching derivation. We may either directly generate a transitive verb treating the relative pronoun as a subject, or we may first perform an $\texttt{N}_\uparrow$-swap first and then generate a transitive verb, treating the relative pronoun as an object. Now the ends of either branch can be labelled to recover our initial examples.

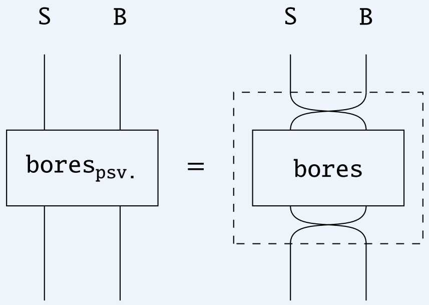


Example 1.3.3 (Passive voice).

$$\texttt{School bores Bob} = \texttt{Bob}\ \underline{\texttt{is bored by}}\ \texttt{school}$$

Twists in wires can be used to model passive voice constructions, which amount to swapping the argument order of verbs. In the original paper CITE , a more detailed analysis including the flanking words `is bored by` involves introducing a new diagrammatic region, which is modelled by having more than a single 0-cell in the *n*-categorical signature.

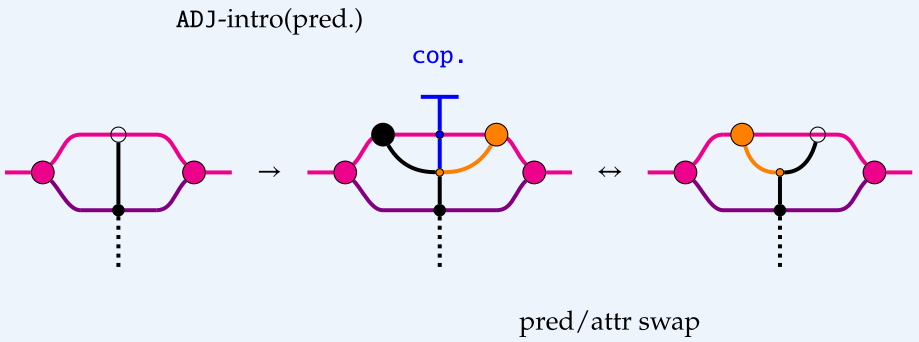




Example 1.3.4 (Copulas).

$$\texttt{Red car} = \texttt{Car is red}$$

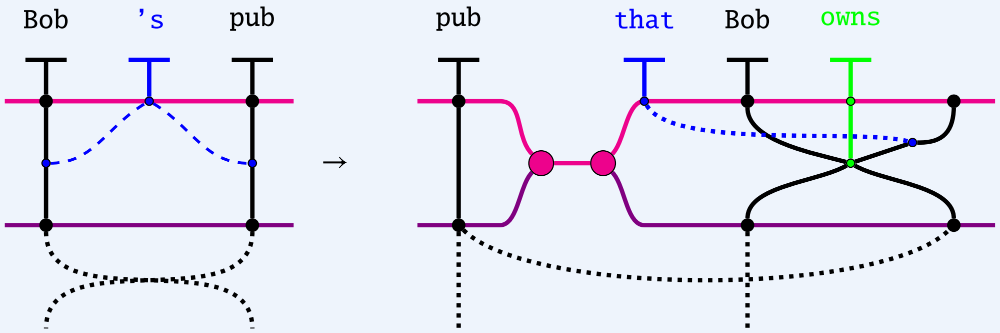
Modifiers such as adjectives and adverbs when they occur before their respective noun or verb are called *attributive*. When modifiers occur after their respective target, they are called *predicative*. In English, without the aid of `and`, only a single predicative modifier is permissible, e.g. `big red car` and `big car is red` are both acceptable, but textttcar is big red is not. There is no issue in introducing rewrites to handle copular modifier constructions in text diagrams, and in text circuits, there is no distinction between either kind of modifier.

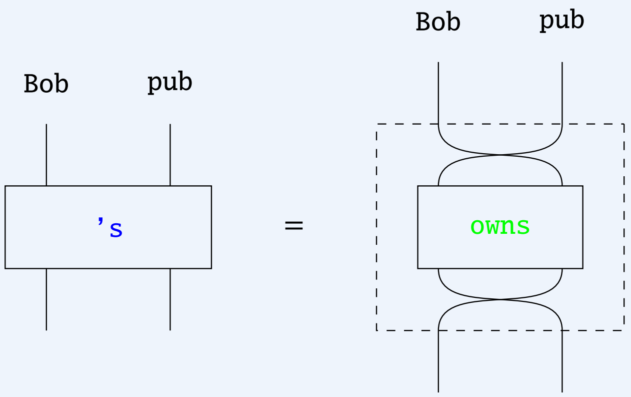


Example 1.3.5 (Possessive pronouns).

$$\texttt{Bob}\underline{\texttt{'s}}\ \texttt{pub} = \texttt{Pub}\ \underline{\texttt{that}}\ \texttt{Bob}\ \underline{\texttt{owns}}$$

This example, along with other grammar equations, was first introduced in the pregroups and internal wirings context in CITE . Possessive pronouns are placed contiguously in between noun-phrases, for which the diagrammatic technology we developed for placing adpositions can be repurposed. Possessive pronouns may be dealt with by a single rewrite that relies on the presence of a transitive ownership verb in the lexicon, which corresponds to a box-analysis in text circuits.

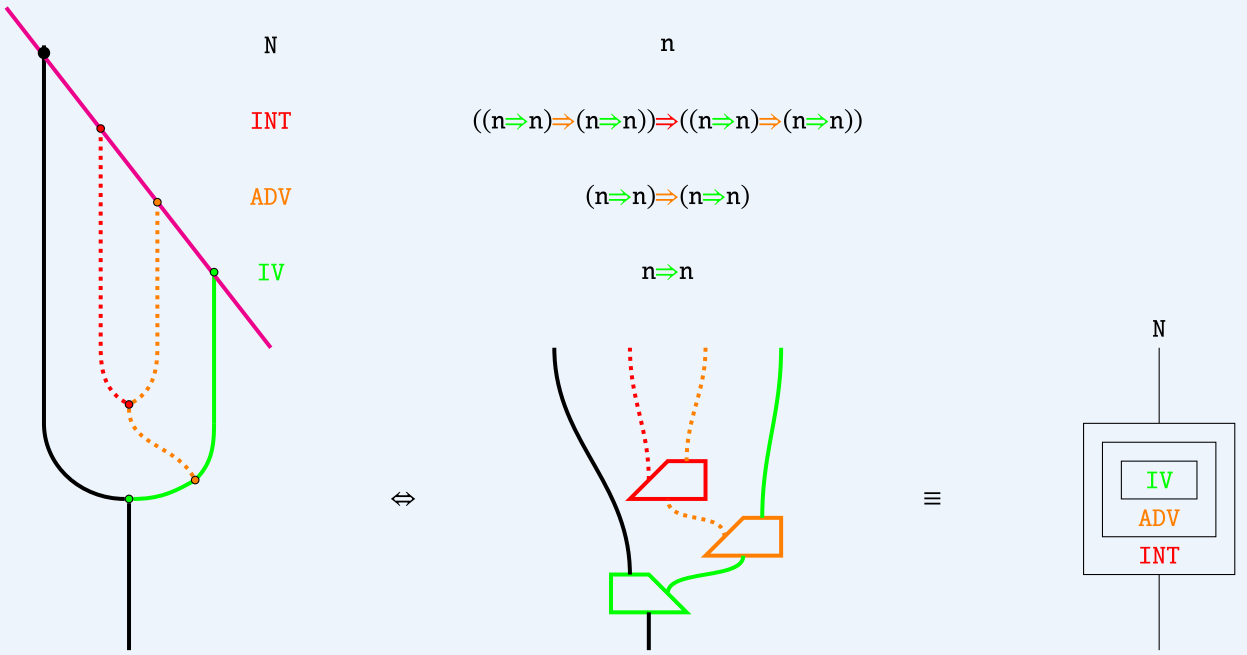




Example 1.3.6 (Intensifers).

`Alice very quickly runs`

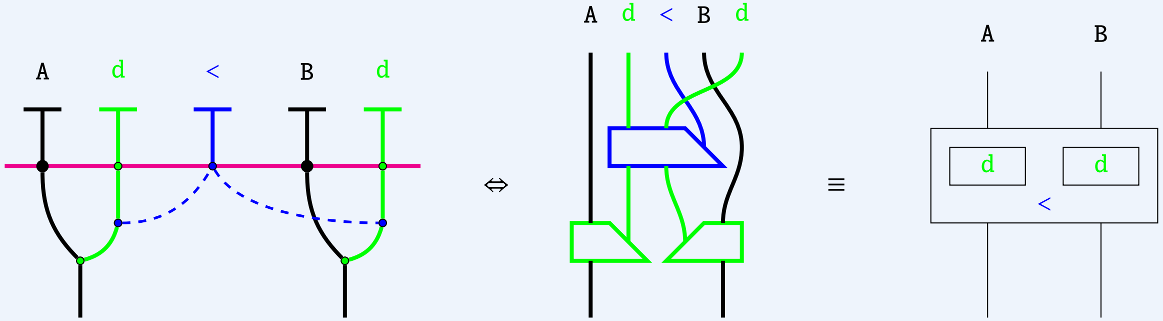
The deep nodes of a text diagram may be equivalently viewed as evaluators in a symmetric monoidal closed setting, and the surface nodes as states for the evaluators. By Curry-Howard-Lambek, this view recovers typelogical grammar settings where composition is some variant of modus ponens. So long as the typing rules are operadic or treelike (which is almost always the case for typelogical grammars, as there are rarely gentzen-style sequent rules that generate multiple outputs), we may instead use a notation where parent edges of evaluation branches become nesting boxes.



Example 1.3.7 (Comparatives).

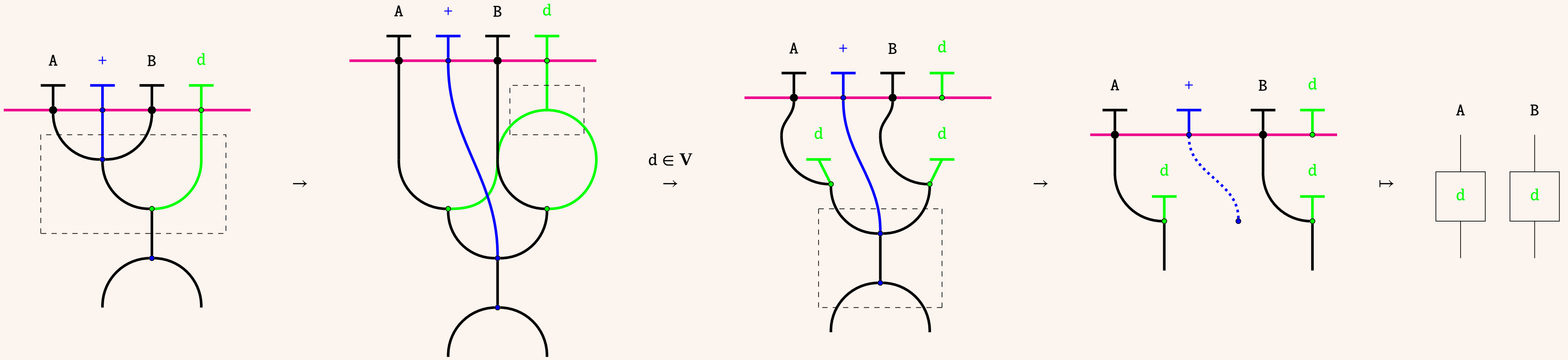
Alice drinks less than Bob drinks

Just as transitive verbs modify two nouns, comparatives are higher-order transitive modifiers that act on the data of verbs or adjectives. A benefit of the symmetric monoidal closed view is that it easily accommodates mixed-order and multi-argument modifiers.



Example 1.3.8 (Syncategorematicity I). *Syncategorematic* words are roughly those that have contextually-dependent semantics. Their dependency is usually predicated on the grammatical type of their arguments. In our terms, since we consider the semantics of text circuits to be underpinned by monoidal functors that reify the circuits in a target category, syncategorematic words such as `and` may be treated as distributive laws. Here `and` occurs as a conjunction of nouns and is eliminated by distributive-law rewrites within the deep structure of the text diagram *before translation into circuits*. Note that what is meant by *distributive* here is, in string-diagrammatic terms, precisely the same as that in algebra, for expressions such as $a \times (b + c) \quad = \quad (a \times b) + (a \times c)$. A new copy-node for verb labels that has rewrites for all verbs facilitates distribution, and the deep `and` nodes come in a tensor-dentensor pair analogous to those for nonstrict string diagrams CITE . Sources of rewrites are outlined in dashed boxes.

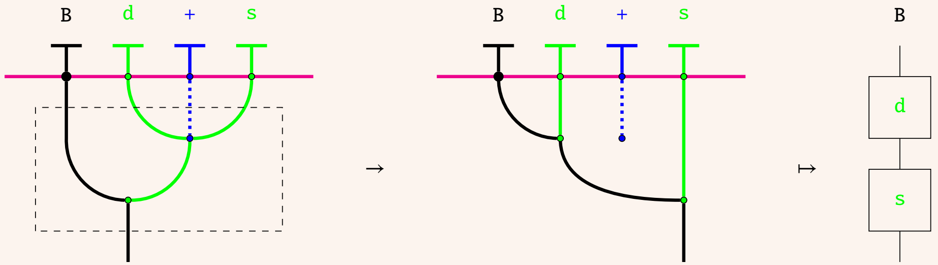
`Alice and Bob drink`



Example 1.3.9 (**Syncategorematicity II**).

`Bob drinks and smokes`

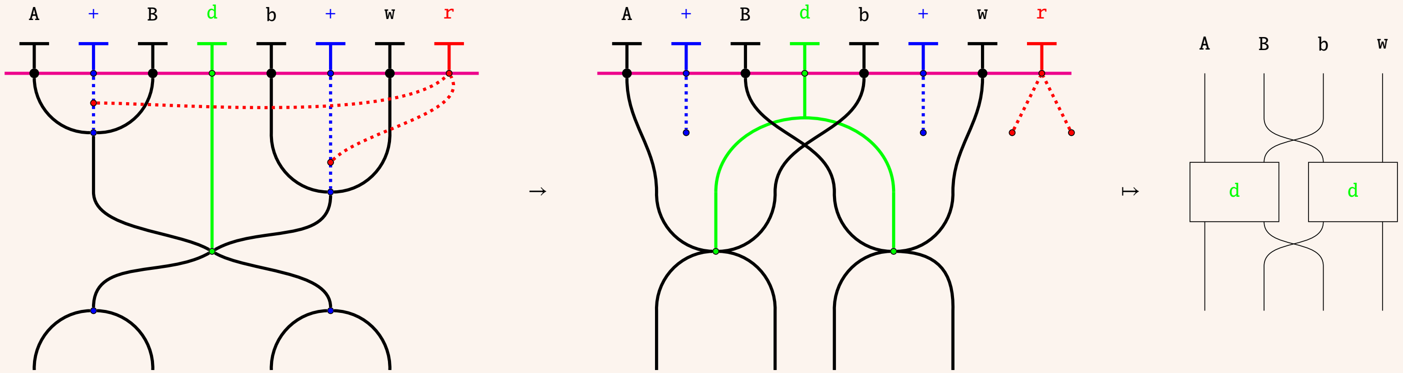
In this example, the same word and is a conjunction of verbs. In this case we choose to interpret the conjunction of verbs as sequential composition, so there is no need for a corresponding detensor for the and of verbs.



Example 1.3.10 (**Coordination**).

`Alice and Bob drink beer and wine respectively`

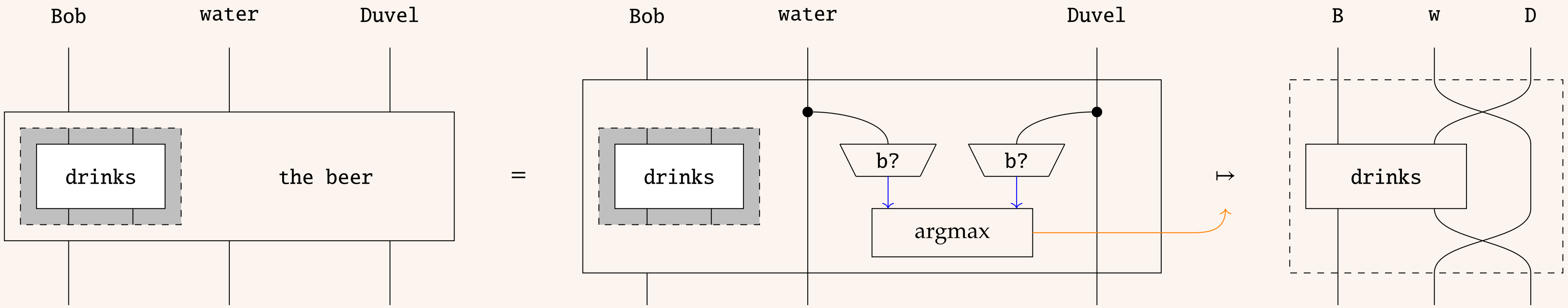
We stand to win in terms of conceptual economy for modelling; more complex phenomena of text structure such as coordination appear to be resolvable in the same framework of distributivity-law rewrites.



Example 1.3.11 (Determiners I).

Bob drinks $\underline{\texttt{the}}$ beer (among drinks)

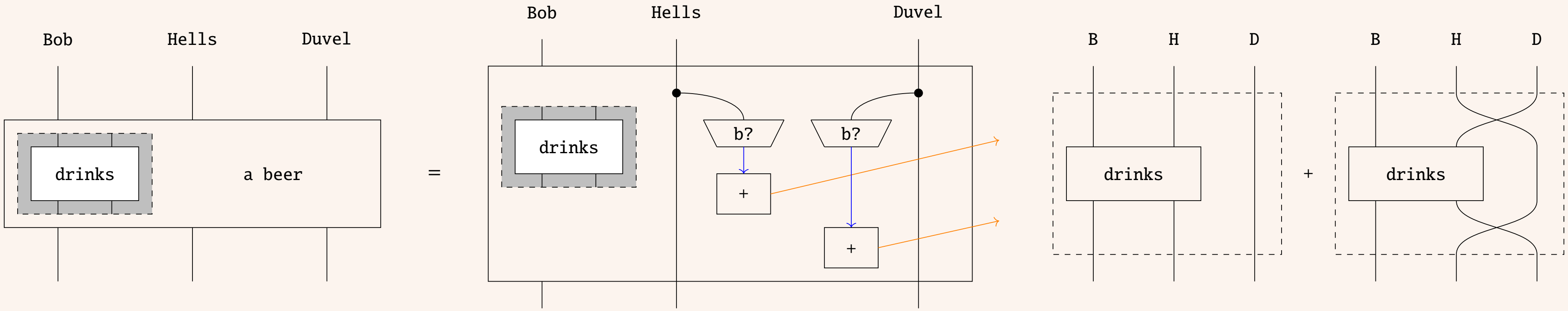
Here, drinks is considered transitive and the beer a nesting box for drinks that reaches over to contextual wires representing a selection of beverages. In this case (relying on the implicit uniqueness of the), a series of beer? tests may be computed, and the best match chosen as the resulting argument for drinks.



Example 1.3.12 (Determiners II).

Bob drinks $\underline{\texttt{a}}$ beer (among drinks)

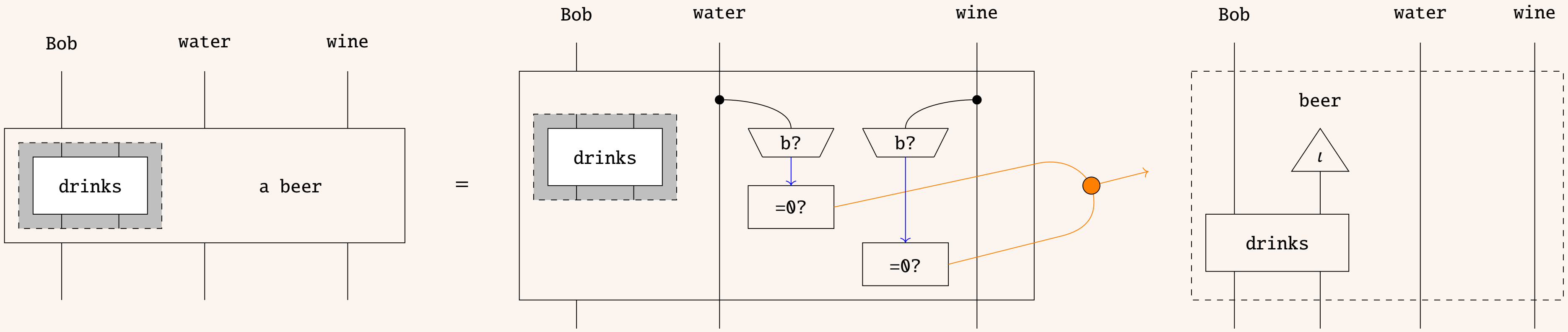
We take the logical (and pragmatic) reading of a as $\exists ! x \ : \ \texttt{beer?}(x) \wedge \texttt{drinks?}(\texttt{Bob}, x)$. Subject to having a method to hold onto alternatives – in essence an inquisitive semantics approach – we may create alternative circuits for each successful beer? test.



Example 1.3.13 (Determiners III).

Bob drinks a beer (that we didn't know about)

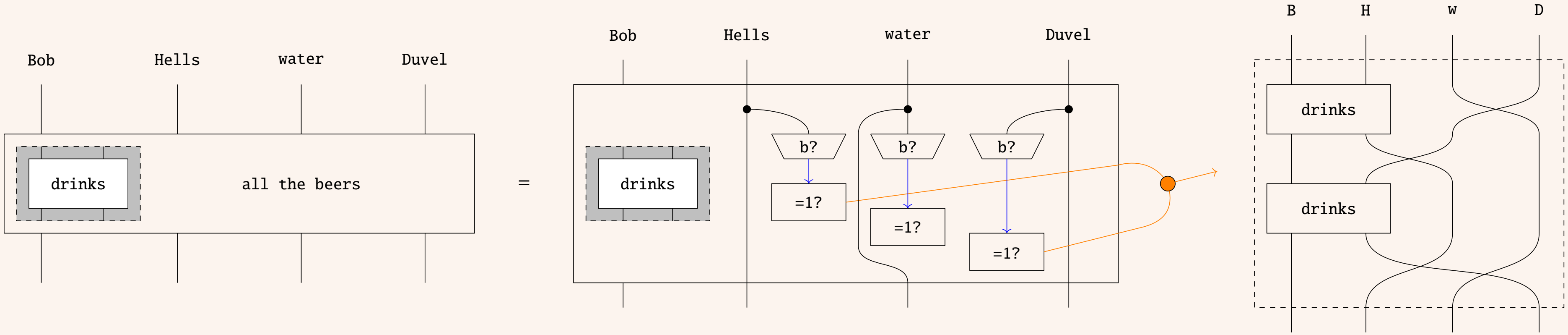
When there are no beers in context, the same statement takes on a dynamic reading: it constitutes the introduction of a beer into discourse. In terms of text circuits, this amounts to introducing a novel beer-state and beer-wire. Determining an appropriate setting to accommodate "arbitrary" vs. "concrete" beers (c.f. Fine's arbitrary objects CITE) requires further research and experimentation, but preliminarily it is known that density matrices are capable of modelling semantic entailment CITE , at the computational cost of adopting the kronecker product. This diagram doesn't typecheck, but note that it doesn't have to, because our strategy for evaluation of determiners treats circuits as syntactic objects to be manipulated.



Example 1.3.14 (**Quantifiers I**).

`Bob drinks` <u>`all the beers`</u> (in context)

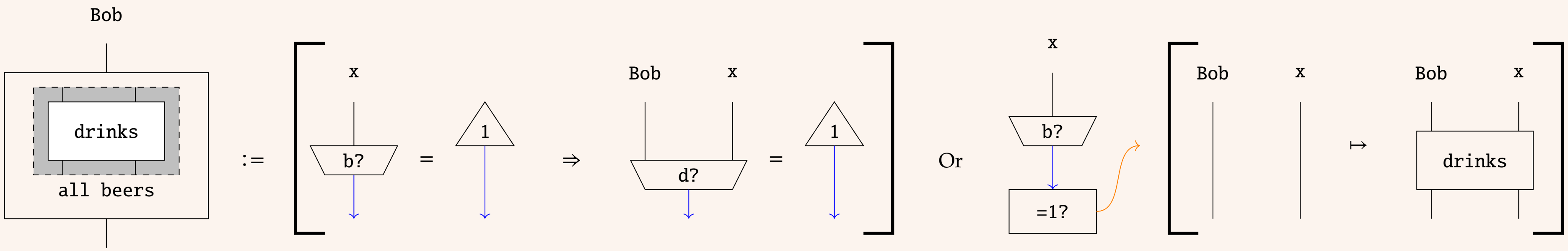
In a finitary context, drinking `all the beers` amounts to applying the distributivity of and iteratively in that context. In this case, `all the beers` is treated as a reference-in-context to `Hells` and `Duvel`. In the same manner, existential quantifiers in finite contexts can be treated as finitary disjunctions, which is handled by creating alternative circuits, as in Example **??**



Example 1.3.15 (**Quantifiers II**).

Bob drinks <u>all</u> beers (generic)

Without the determiner the, this becomes a generic statement, which logically amounts to (analysing the usual conditional as a disjunction) $\forall x : \neg\texttt{beer?}(x) \vee \texttt{drinks?}(\texttt{Bob}, x)$. We can treat generic universal quantifiers of this kind in at least two ways. The first essentially truth-conditional approach is to treat the generic as a process-theoretic condition governing measurements: whenever it is the case that something is a beer, it is the case that Bob drinks it. The second "inferential" appraoch is to treat the generic as a rewrite of text circuits conditioned on a beer test: whenever something is a beer we may add on a gate witnessing that Bob drinks that beverage.


Bob
drinks
all beers
:=
x
b?
=
1
⇒
Bob
x
d?
=
1
Or
x
b?
=1?
Bob
x
↦
Bob
x
drinks