REWRITING STRUCTURED COSPANS

DANIEL CICALA

Abstract. To foster the study of networks on an abstract level, we further study the formalism of structured cospans. We define a topos of structured cospans and establish its theory of rewriting. For the rewrite relation, we propose a double categorical semantics to encode the compositionality of the structure cospans. For an application, we generalize the inductive viewpoint of graph rewriting to rewriting in a wider class of topoi.

(

- systematic use of emph, '' and " " and bold face
- RGRAPH vs GRAPH? To evangelize the poisitoin of RGRAPHs superiority in CT over GRPH or not?

)

1. Introduction

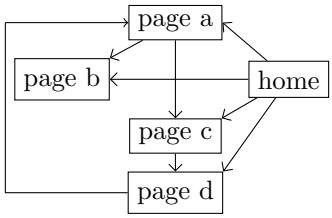
Structured cospans are a framework for reasoning about systems with inputs and outputs. Rewriting is a topic that covers methods for editing substructures of an object such as a string or a graph. In this paper, we introduce rewriting to structured cospans.

While the term "system" used above is intentionally vague, a first approximation of what we have in mind is a collection of discrete objects that are somehow related. Examples include home electrical systems consisting of appliances connected by wires; social systems where people are connected by social relationships; or virtual systems such as a collection of webpages connected by links. These systems are often analyzed as though they are isolated, that they are *closed systems*. Indeed, one considers a single home, or a particular clique of people, or a single webpage, entirely ignoring that each of these are actually part of a larger system. A home lives in power grid, a clique is a merely one portion of the human population, and a webpage is actually housed in the internet. That is, these are really *open systems* that can interact with the other compatible systems. Structured cospans offer a way to equip closed systems with the mechanisms through which these interactions can occur.

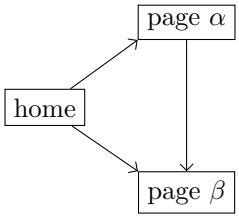
The following toy example suggests a way to fit a system into its larger context. Suppose we want to analyze the structure of Professor Smith's academic website, which contains a home page and two sections. The section devoted to research

Date: January 15, 2020.

comprises, say, five pages connected by several links as modelled by the graph

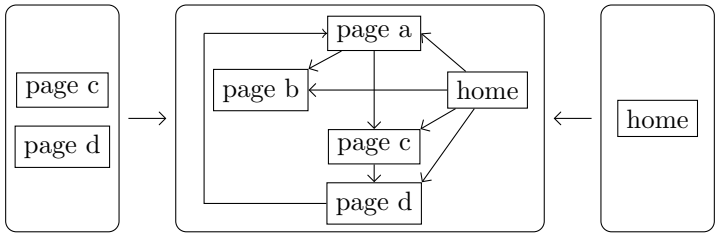


The section devoted to teaching comprises, say, three pages connected by several links as modelled by the graph



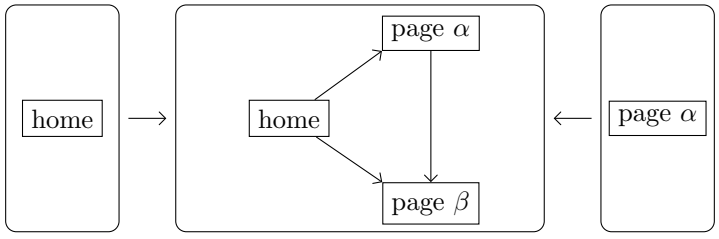
These graphical models are amenable to analysis using tools from network theory. However, such graphical models fail to capture that Professor Smith's website is part of the larger internet, rather, that it is an open system. This leads to us replacing each graph in our model with an *open graph*, meaning a graph equipped with two sets of nodes interpretted as 'inputs' and 'outputs' (terms not meant to imply causality). We can promote a graph to an open graph by equipping it with a pair of functions from discrete graphs that effectively select the inputs and outputs. For instance

(1)



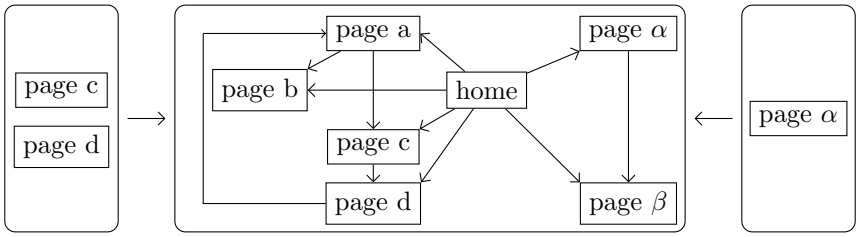
gives the graph inputs 'page b' and 'page c' and outputs 'home'. Likewise

(2)



gives the graph inputs 'home' and outputs 'page α'. These cospans provide a mechanism to connect compatible systems together, thus allowing us to see how a system fits into its surroundings. Observe that the outputs of (1) match the inputs of (2). This indicates that the two sections of the website share a page, namely the home page. It also means that we can model the entire website by connecting the

two graphs along that common node



This example can be formalized using structured cospans.

Baez and Courser introduced structured cospans as an abstract framework to model open systems [1]. A structured cospan is a diagram of the form

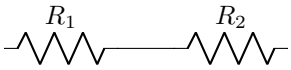
$$La \to x \leftarrow Lb \tag{3}$$

where $L\colon \mathsf{A} \to \mathsf{X}$ is a functor whose codomain X has pushouts. This functor is a nice bookkeeping device that allows us to separate the system types into X and the interface types into A. We then interpret (3) as a system x with inputs La and outputs Lb. There is a category ${}_L\mathsf{Csp}$ whose objects are those of A and arrows are (isomorphism classes of) structured cospans $La \to x \leftarrow Lb$ that compose by pushout:

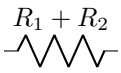
$$(La \to x \leftarrow Lb \to y \leftarrow Lc) \mapsto (La \to x +_{Lb} y \leftarrow Lc)$$

In the example above, $\mathsf{A} \coloneqq \mathsf{Set}$, $\mathsf{X} \coloneqq \mathsf{Graph}$, and L turns a set into a discrete graph.

Given the ability to model open systems using structured cospans, we would like tools to analyze these models. In this paper, we adapt for structured cospans the tool of double pushout (DPO) rewriting, an algorithmic technique for creating a new object from an old object according to some given set of rules. One application of rewriting is to "simplify" objects. For example, in the case of electrical circuits, rewriting provides a method to replace an instance of



in a circuit diagram with

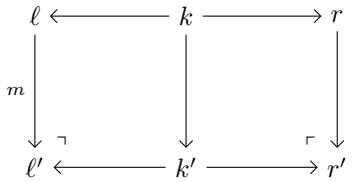


Rewriting works nicely in any adhesive category [8]. However, we restrict our attention to a specific type to adhesive category called a topos [9]. The reasons for this are both technical (we eventually required cartesian closedness) and promotional (topoi are more widely known and are more easily understood than adhesive categories).

In DPO rewriting, one starts with a *grammar* (T, P), that is an topos T and a set $P \coloneqq \{\ell_j \leftarrow k_j \to r_j\}$ of spans in T with monic arrows. These spans are called *rules*. We interpet a rule $\ell \leftarrow k \to r$ as stating that r replaces ℓ in a manner that fixes k. This rule can be applied to any object ℓ' by realizing a *double pushout*

diagram

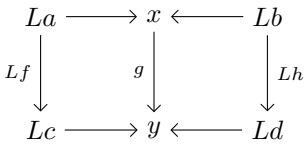
(4)



where m identifies an instance of ℓ in ℓ' and that instance is then replaced by r, thus resulting in the new object r'. The squares being pushouts ensure that the glueing condition is met, which in the case of graphs, means that there are no nodes removed that leave an edge unanchored.

To study a grammar (T, P), one studies its rewrite relation. Assuming that P contains the rule $\ell \leftarrow k \rightarrow r$, the span across the bottom of (4) is called a *derived rule*. Collecting all derived rules $\ell' \leftarrow k' \rightarrow r'$ associated to (T, P), we define a relation $\ell' \rightsquigarrow r'$. The reflexive and transitive closure $\rightsquigarrow^*$ is called the *rewrite relation* for (T, P). The meaning of $x \rightsquigarrow^* y$ is that there is a finite sequence of rules in P through which x can be rewritten into y. In some sense, y is a simplification of x. From (T, P) and its associated rewrite relation, we can build a category $\mathrm{Lang}(\mathsf{T}, P)$ called the *language* of (T, P). Often, the terms "language" and "rewrite relation" are used interchangably, but we will reserve the former for the category theoretical persepctive and the latter for the relational perspective. The category $\mathrm{Lang}(\mathsf{T}, P)$ has the same objects as T and the arrows $x \rightarrow y$ witness $x \rightsquigarrow^* y$. In fact, by defining a category Gram of grammars, Lang is seen to be functorial $\mathrm{Lang}\colon \mathsf{Gram} \rightarrow \mathsf{Cat}$ (Theorem 2.2) thus encoding the rewrite relation in a category.

In order to bring structured cospans into the theory of rewriting, we show that they form a topos. This is true under certain conditions. Define a category ${}_L\mathsf{StrCsp}$ whose objects are structured cospans and arrows are commuting diagrams

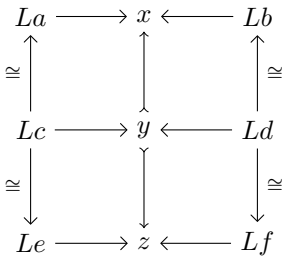


If L is a geometric morphism, that is a left exact left adjoint between topoi, then ${}_L\mathsf{StrCsp}$ is a topos because is it equivalent to the Artin glueing $\mathsf{A} \times \mathsf{A} \downarrow \Delta R$ where R is right adjoint to L and $\Delta\colon \mathsf{A} \rightarrow \mathsf{A} \times \mathsf{A}$ is the diagonal functor.

Even though we are restricting L more than Baez and Courser, our definition still covers many important examples. One such example is the discrete graph functor $L\colon \mathsf{Set} \rightarrow \mathsf{Graph}$ mentioned above. More examples come by using slice categories Graph/g for some graph g chosen to endow nodes and arrows with types as was done to model the ZX-calculus [3].

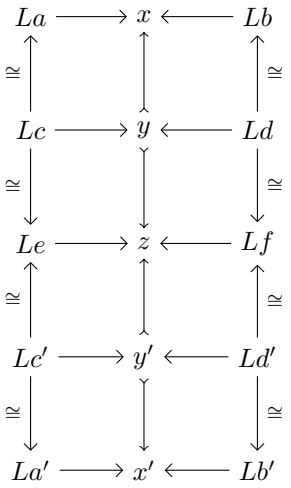
Because ${}_L\mathsf{StrCsp}$ is a topos, it supports a rich theory of rewriting. An analysis begins with a *structured cospan grammar* $({}_L\mathsf{StrCsp}, P)$, which is different than a grammar as discussed above because we require P to contain spans in ${}_L\mathsf{StrCsp}$ of

the form

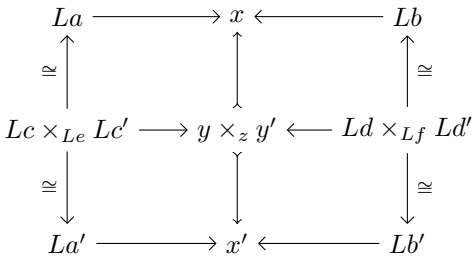


This condition is stronger than simply requiring monic-legged spans due to the isomorphisms present. These isomorphisms ensure that rewriting cannot change the interface of a system.

Given a structured cospan grammar, we can construct its language in a functorial way. This starts with an observation: there are two compositional structures at play. There is the compositionaliy of the rules, seen by placing two atop one another

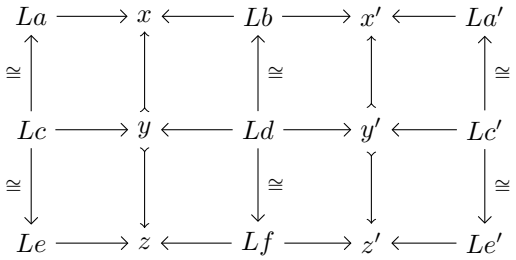


and obtaining the composite

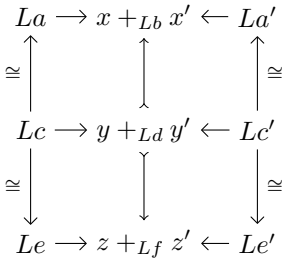


using pullback. In classical rewriting, this is the only composition. New to structured cospan rewriting is the compositionality of the structured cospans themselves,

seen by placing rules beside one another



and composing using pushouts



Because there are two compositional structures, encoding the language of a structured cospan grammar into just a category is insufficient. Evidently, we should instead use a double category.

The language functor for structured cospans has type

$$\mathrm{Lang}\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$$

where the domain is a category whose objects are structured cospan grammars and codomain is the category of double categories (Theorem 4.5). Through this functor, we assign to each structured cospan grammar $({}_L\mathsf{StrCsp}, P)$ a double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ serving as its language. The horizontal arrows of this language are structured cospans. The squares are generated by the rules derived from P. Manifestly, if one structured cospan can be rewritten into another using rules in P then there is a square between them. In this sense, $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ encodes the rewrite relation $\rightsquigarrow^*$. Although, there is a new manner of deriving rules through the compositionality of the structured cospans. And so, a theory of rewriting is established.

Further work is required to study properties of the languages arising from structured cospan grammars. Now, we turn our attention to an application of rewriting structured cospans.

In the classical topics of rewriting, formal languages and term rewriting, there are two approaches to defining the rewrite relation for a grammar. The first is an operational definition which stipulates when a rule can be applied by using subterms and substitution. The other is an inductive definition which constructs the rewrite relation using generators and closure operations. When rewriting theory expanded to graphs in the 1970's, only the operational definition prevailed. Then in the 1990's, Gadducci and Heckel introduced an inductive definition to graph rewriting [6], thus allowing for analyses using structural induction. With the new

technology of structured cospans, we can use their ideas to bring the inductive viewpoint to rewriting in a large class of topoi.

A central idea in developing the inductive definition is to equip graphs with an interface. Earlier, we refered to such graphs as open, but Gadducci and Heckel called them "ranked graphs". To bring this idea to objects of a topos, we can use structured cospans. That is, if X is a topos that fits into a geometric morphism $L \dashv R\colon \mathsf{X} \to \mathsf{A}$ with a monic counit, then an *object x with an interface* is a structured cospan $La \to x \leftarrow Lb$. The induced comonad on X can be though of as returning the maximal, with respect to inclusion, interface LRx of each object x. The monic counit $LRx \to x$ simply includes that interface.

Another pillar in the construction of the inductive definition is the equivalence between two classes of rewrite relations. In the context of graph rewriting, this result states that the rewrite relation for a graph grammar

$$(\mathsf{Graph}, \{\ell_j \leftarrow k_j \to r_j\})$$

is the same as for the 'discrete' graph grammar

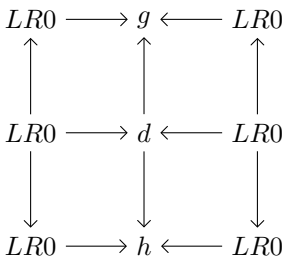
$$(\mathsf{Graph}, \{\ell_j \leftarrow k'_j \to r_j\})$$

where k'_j is the discrete graph underlying k_j. To extend this to rewriting in a topos, we again use a comonad LR induced from a geometric morphism with a monic counit. It allows us to access the "discrete" objects LRx underlying an arbitrary object x. In the generalized setting, the rewrite relation for the grammar $(\mathsf{X}, \{\ell_j \leftarrow k_j \to r_j\})$ is the same as for the grammar $(\mathsf{X}, \{\ell_j \leftarrow LRk_j \to r_j\})$ as long as the subobject lattices $\mathrm{Sub}(k_j)$ have all meets (Theorem 5.6).

We now have the ability to equip the objects of a topos with an interface using structured cospans and we have established that a grammar and its "discretized" version induce the same rewrite relation. Next, we can provide an inductive definition for the language. Fix a grammar $(\mathsf{X}, \{\ell_j \leftarrow k_j \to r_j\})$ such that X fits into an adjunction $L \dashv R\colon \mathsf{X} \to A$ with a monic counit and, for each j, $\mathrm{Sub}(k_j)$ has all meets. Using this data, we construct a sub-double-category of ${}_L\mathsf{StrCsp}$ generated on squares

$$\begin{array}{ccccc}
L0 & \longrightarrow & \ell_j & \longleftarrow & LRk_j \\
\uparrow & & \uparrow & & \uparrow \\
L0 & \longrightarrow & LRk_j & \longleftarrow & LRk_j \\
\downarrow & & \downarrow & & \downarrow \\
L0 & \longrightarrow & r_j & \longleftarrow & LRk_j
\end{array}$$

Then, $g \rightsquigarrow^* h$ if and only if there is a square



in our double category (Theorem 6.2). Because we have completely characterized the language with these squares, the generating squares provide the inductive definition.

1.1. **Outline and contributions.** In Section 2, we review double pushout rewriting in topoi. This section culminates in a functorial construction of the rewrite relation. The author is currently unaware of any such construction in the literature.

In Section 3, we introduce a modified definition of structured cospans. They are placed as arrows into the category ${}_L\mathsf{Csp}$ (as done by Baez and Courser) and we introduce a new category ${}_L\mathsf{StrCsp}$ in which they are objects. The main result of this section is that ${}_L\mathsf{StrCsp}$ is a topos (Theorem 3.5) that is constructed functorially in L (Theorem 3.6).

The theory of double pushout rewriting is introduced to structured cospans in Section 4. This does not entail simply restricting the theory of rewriting in a topos to structured cospans, but requires accommodations to the added compositional structure. The layout of this section echos the review of DPO rewriting in Section 2. We give a functorial construction of the language for a structured cospan grammar using double categories (Theorems 4.5 and 4.6).

In their foundational paper on graph rewriting, Ehrig, Pfender, and Schneider classify the expressivity of several types of grammars [5, Prop. 3.3]. We generalize this result to certain grammars on topoi in Section 5 (Theorem 5.6).

Finally, in Section 6, we use rewriting structured cospans to provide an inductive viewpoint of rewriting in topoi (Theorem 6.2).

1.2. **On exposition.** In this paper, we move through three levels of abstraction. The most abstract involves working in a topos. The intermediate level involves working with the notion of a system that can connect together with other compatible systems. By a system, we mean a collection of entities that are somehow related. Examples include social systems, electrical systems, physical systems, etcetera. At our least abstract, we work with a particular system, such as the internet, which serves as a running example throughout.

The reason we speak in terms of systems is to make clear the motivation for structured cospans: adding compositionality to network theory. We jump between these three levels of abstraction in a casual, but deliberate manner. We make this explicit to help the reader navigate their way through the exposition.

1.3. **Acknowledgements.** The author would like to thank John Baez for the many helpful conversations during the preparation of this paper.

2. Rewriting in topoi

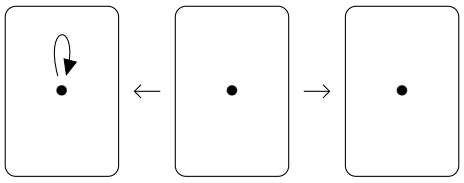
A common tool to model a system of objects that are somehow related is the graph. A graphical model offers a rich theory to any system's analysis. One fruit of this theory, called double pushout (DPO) graph rewriting, provides an algebraic method to determine whether two systems are semantically equivalent. An example from electrical engineering is the equivalence between two resistors R_1 and R_2 wired in series and a single resistor $R_1 + R_2$.

Double pushout rewriting has an established literature, so we use the current section to cover the fundamentals and to establish our conventions. The interested reader can see Ehrig, et. al. [5] to learn about graph rewriting or Lack and Sobociński [8] for an axiomatic approach based on *adhesive categories*. Here, instead of working with the full generality offered by adhesivity, we work inside of a topos. This allows us to reap the benefits from a rich rewriting theory because topoi are adhesive [9], while retaining the technical requirements for our constructions to work. Henceforth, every category is a topos.

Rewriting starts with the notion of a **rewrite rule**, or simply **rule**. This is a span $\ell \leftarrow k \rightarrow r$ with two monic arrows. The interpretation of this rule is that ℓ can be replaced by r and k is the part of ℓ that does not change.

For example, suppose we wanted to enumerate paths via links between two pages of a website. One way to do this is to model the internet as a graph where nodes correspond to websites and edges to links. In particular, our graph should have no self-loops. Otherwise, two nodes would either be connected by no paths or by infinitely many paths, thus rendering counting useless. To remove loops, we can introduce the rule

(5)



Typically, a collection of rewrite rules is packaged together with a given system. Resistor circuits have parallel, series, and star rules. Word processors replace misspelled words with their correction. Calculators replace the string $2 + 2$ with the string 4. We formalize this idea with the following definition.

Definition 2.1 (Grammar)**.** A **grammar** is a topos T together with a finite set of rules $P \coloneqq \{\ell_j \leftarrow k_j \rightarrow r_j\}$. A morphism of grammars $(\mathsf{S}, P) \rightarrow (\mathsf{T}, Q)$ is a pullback and pushout preserving functor $F \colon \mathsf{S} \rightarrow \mathsf{T}$ such that Q contains the image of P. These form a category Gram.

Returing to our model of the internet, we might consider the grammar (Graph, P) where P contains a single rule: Rule (5). We can apply this rule to suitable objects of Graph. For instance, given a graph g with a self-loop, we can apply our rule to g and produce a new graph: g with the loop removed.

What, precisely, do we mean when we say "apply"? In general, we can *apply* a rule $\ell \leftarrow k \rightarrow r$ to an object ℓ' using any arrow $m \colon \ell \rightarrow \ell'$ for which there exists a

pushout complement, that is an object k' fitting into a pushout diagram

$$\begin{array}{ccc} \ell & \longleftarrow & k \\ {\scriptstyle m}\downarrow & \lrcorner & \downarrow \\ \ell' & \longleftarrow & k' \end{array}$$

A pushout complement need not exist, but if it does and the map $k \to \ell$ is monic, then it is unique up to isomorphism [8, Lem. 15].

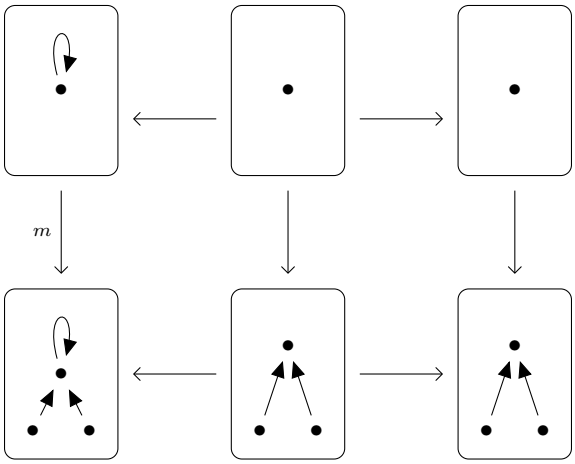
Every application of a rule begets a new rule. Applying $\ell \leftarrow k \to r$ to ℓ' along $m\colon \ell \to \ell'$ induces a **derived rule** $\ell' \leftarrow k' \to r'$ obtained as the bottom row of the double pushout diagram

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ {\scriptstyle m}\downarrow & \lrcorner & \downarrow & \llcorner & \downarrow \\ \ell' & \longleftarrow & k' & \longrightarrow & r' \end{array}$$

This diagram expresses a three-stage process whereby m selects a copy of ℓ inside ℓ', this copy is replaced by r, and the resulting object r' is returned. Because pushouts preserve monics in a topos, a derived rule is, in fact, a rule.

Let us illustrate this using our graphical model of the internet. Rule (5) is applicable to any node with a loop, such as in the double pushout diagram

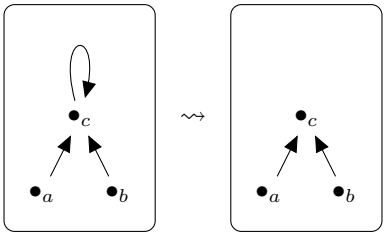
(6)



In the bottom left graph, we have modeled a portion of the internet where one webpage has a link to itself. By applying Rule (5) to this graph, we obtain a model of this same portion of the internet minus this link. If we were counting paths between webpages, the new model is preferable.

A grammar (T, P) induces a collection dP of all derived rules obtained by applying a rule in P to an object in T. We can use dP to analyze the grammar (T, P) by constructing the *rewrite relation* $\rightsquigarrow^*$. The meaning of $x \rightsquigarrow^* y$ is that we can rewrite x into y by appling a sequence of rules in dP. To precisely define the rewrite relation, we start by constructing a relation $\rightsquigarrow$ on the objects of T by setting $\ell' \rightsquigarrow r'$ if there exists a rule $\ell' \leftarrow k' \to r'$ in dP. For instance, Diagram (6)

implies that



However, $\rightsquigarrow$ does not capture enough information about (T, P), which is why we define the **rewrite relation** $\rightsquigarrow^*$ to be the reflexive and transitive closure of $\rightsquigarrow$.

The rewrite relation can be encoded as arrows in a category via a functorial construction we call the "language functor". In the rewriting literature, the terms "language" and "rewrite relation" are interchangable. However, we give them sligthly different meanings in order to help orient the reader. Namely, we use "rewrite relation" when giving a relational perspective and "language" when giving the category theoretical perspective.

Theorem 2.2. *Let (T, P) be a grammar and dP be the set of all rules derived from (T, P). Define a relation $\rightsquigarrow$ on objects of T by $\ell' \rightsquigarrow r'$ if and only if there is a rule $\ell' \leftarrow k' \rightarrow r'$ in dP.*

There exists a category $\mathrm{Lang}(\mathsf{T}, P)$ whose objects are those of T and arrows are generated by the relation $x \rightsquigarrow y$. Given a morphism of grammars $F\colon (\mathsf{T}, P) \to (\mathsf{T}', P')$, there is a functor $\mathrm{Lang}(F)\colon \mathrm{Lang}(\mathsf{T}, P) \to \mathrm{Lang}(\mathsf{T}', P')$ defined on objects by $x \mapsto Fx$ and on arrows by extending from $(x \rightsquigarrow y) \mapsto (Fx \rightsquigarrow Fy)$. This defines a functor $\mathrm{Lang}\colon \mathsf{Gram} \to \mathsf{Cat}$

Corollary 2.3. *Let $\rightsquigarrow^*$ be the rewrite relation for (T, P) and let x and y be objects of T. There is an arrow $x \to y$ in $\mathrm{Lang}(\mathsf{T}, P)$ if and only if $x \rightsquigarrow^* y$.*

Proof. Sufficency follows by construction. For necessity, suppose $x \rightsquigarrow^* y$. Then either $x = y$ and the identity arrow in $\mathrm{Lang}(\mathsf{T}, P)$ is the arrow we seek, or there is a sequence $x \rightsquigarrow x_1 \rightsquigarrow \cdots \rightsquigarrow x_n \rightsquigarrow y$ which gives the sequence of arrows $x \to x_1 \to \cdots \to x_n \to y$ in $\mathrm{Lang}(\mathsf{T}, P)$ whose composite is the arrow we seek. □

Though there is more to the theory of rewriting that providing in this section, we have developed enough of the theory to continue our goal of introducing rewriting to structured cospans.

3. Structured cospans form a topos

Every topos supports a rich rewriting theory. This fact underpins our efforts to introduce rewriting to structured cospans. And so, in this section, we reintroduce the notion of structured cospans and find sufficient conditions for them to form a topos.

Baez and Courser [1] introduced structured cospans as a framework to study open systems. A system is **open** when equipped with a mechanism by which it can connect to any compatible system. For example, a vacuum cleaner can connect with the electrical grid via an electrical socket. A pulley system can connect to a mechanical motor. An open system stands in constract to a closed system that cannot interact with its outside environment.

In order to rewrite structured cospans in the sense of Section 2, they must form a topos. To acheive this, we impose stronger conditions than given by Baez and Courser. To specify these conditions, we use what is known as a geometric morphism. This is an adjunction

$$\mathsf{X} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{A}$$

between topoi with L left exact.

Definition 3.1. Fix a geometric morphism $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. An **$L$-structured cospan**, or simply **structured cospan**, is a diagram $La \to x \leftarrow Lb$ in X.

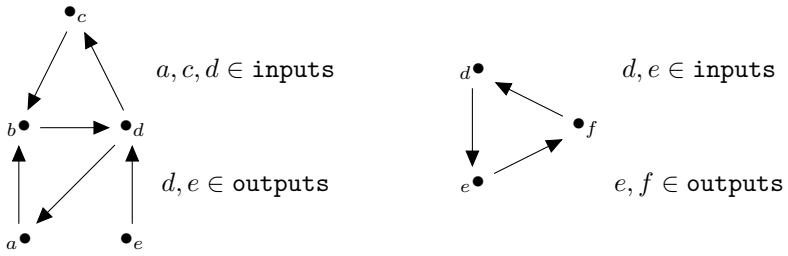
To interpret $La \leftarrow x \to Lb$ as an open system, take x to represent the system with inputs La chosen by the arrow $La \leftarrow x$ and outputs Lb chosen by $Lb \to x$. This open system can now connect to any other open system with inputs Lb, say $Lb \leftarrow y \to Lc$. We form the *composite* of the two open systems by connecting x to y along their common interface Lb. Mathematically, this amounts to taking the pushout of x and y over Lb, thus giving the composite system $La \leftarrow x +_{Lb} y \to Lc$.

We intentionally chose the term "composite system" to suggest that structured cospans are the arrows of some category. Indeed, this is the case.

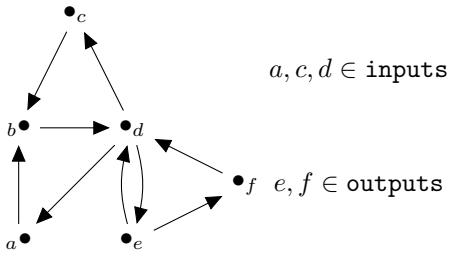
Definition 3.2. Fix a geometric morphism $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. Denote by ${}_L\mathsf{Csp}$ the category whose objects are those of A and whose arrows from a to b are (isomorphism classes of) structured cospans $La \to x \leftarrow Lb$.

For ${}_L\mathsf{Csp}$ to form a category, we do not require the full strength of the conditions listed in Definition 3.2—Baez and Courser do not ask this much—but remember, we are aiming to form a topos of structured cospans.

Example 3.3. Structured cospans can be used to define open graphs. A graph is open when equipped with two subsets of nodes, one set serving as inputs and the other as outputs. When the inputs of one open graph coincide with the outputs of another, they can be composed. For example, the pair of open graphs



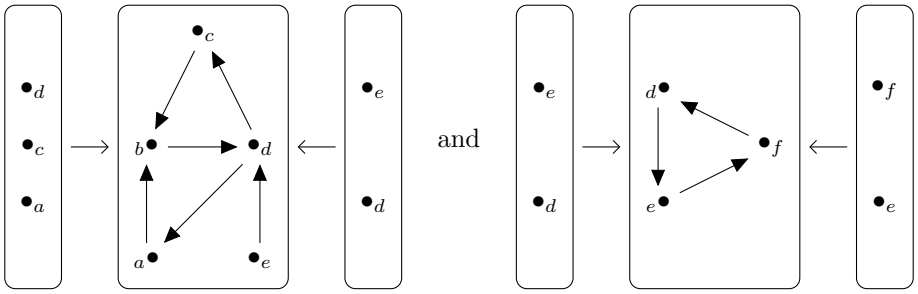
compose by glueing the corresponding nodes together, forming the new open graph



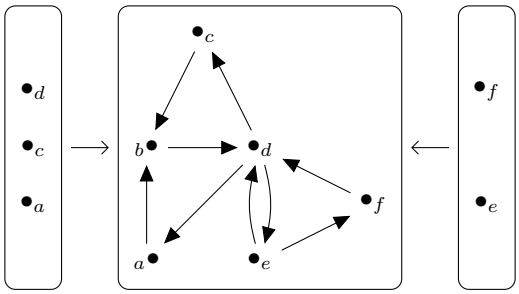
To define an open graph as a structured cospan, consider the geometric morphism

$$\mathsf{Graph} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{Set}$$

where L is the discrete graph functor and R forgets the graph edges. The above open graphs can be presented as the structured cospans



with composite



In this example, the category ${}_L\mathsf{Csp}$ has sets for objects and open graphs for arrows.

Recall that, in Section 2, we saw that rewriting operates on the objects of a topos, not the arrows. Therefore, we cannot hope to rewrite structured cospans inside the category ${}_L\mathsf{Csp}$. Our task, now, is to define a category where structured cospans are objects. Then we can show that category to be a topos.

Definition 3.4. Let $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ be geometric morphism. Define ${}_L\mathsf{StrCsp}$ to be the category whose objects are L-structured cospans and arrows from $La \to x \leftarrow Lb$ to $Lc \to y \leftarrow Ld$ are triples of arrows (f, g, h) fitting into the commuting diagram

$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ {\scriptstyle Lf}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle Lh} \\ Lc & \longrightarrow & y & \longleftarrow & Ld \end{array}$$

Theorem 3.5. *Let $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ be a geometric morphism. The category ${}_L\mathsf{StrCsp}$ is a topos.*

Proof. By adjointness, ${}_L\mathsf{StrCsp}$ is equivalent to the category whose objects are cospans of form $a \to Rx \leftarrow b$ and morphisms are triples (f, g, h) fitting into the commuting diagram

$$\begin{array}{ccccc} w & \longrightarrow & Ra & \longleftarrow & x \\ {\scriptstyle f}\downarrow & & {\scriptstyle Rg}\downarrow & & {\scriptstyle h}\downarrow \\ y & \longrightarrow & Rb & \longleftarrow & z \end{array}$$

This, in turn, is equivalent to the comma category $(\mathsf{A}\times\mathsf{A} \downarrow \Delta R)$ where $\Delta\colon \mathsf{A} \to \mathsf{A}\times\mathsf{A}$ is the diagonal functor, a right adjoint. Because ΔR is a right adjoint, $(\mathsf{A}\times\mathsf{A} \downarrow \Delta R)$ is an Artin glueing [10], therefore a topos. □

Not only is ${}_L\mathsf{StrCsp}$ a topos, but it is constructed functorially.

Theorem 3.6. *Denote by* Topos *the category of topoi and geometric morphisms. There is a functor*

$${}_{(-)}\mathsf{StrCsp}\colon [\bullet \to \bullet, \mathsf{Topos}] \to Topos$$

defined by

$$\begin{array}{ccc} \mathsf{X} & \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\scriptstyle\perp}}}} & \mathsf{A} \\ F\uparrow\dashv\downarrow G & & G'\downarrow\vdash\uparrow F' \\ \mathsf{X}' & \underset{L'}{\overset{R'}{\underset{\longleftarrow}{\overset{\longrightarrow}{\scriptstyle\top}}}} & \mathsf{A}' \end{array} \quad \xmapsto{{}_{(-)}\mathsf{StrCsp}} \quad {}_L\mathsf{StrCsp} \underset{\Theta'}{\overset{\Theta}{\underset{\longleftarrow}{\overset{\longrightarrow}{\scriptstyle\perp}}}} {}_{L'}\mathsf{StrCsp}$$

which is in turn given by

$$\begin{array}{ccccc} La & \xrightarrow{m} & x & \xleftarrow{n} & Lb \\ {\scriptstyle Lf}\downarrow & & {\scriptstyle g}\downarrow & & {\scriptstyle Lh}\downarrow \\ Lc & \xrightarrow{o} & y & \xleftarrow{p} & Ld \end{array} \quad \xmapsto{\Theta} \quad \begin{array}{ccccc} L'G'a & \xrightarrow{Gm} & Gx & \xleftarrow{Gn} & L'G'b \\ {\scriptstyle L'G'f}\downarrow & & {\scriptstyle Gg}\downarrow & & {\scriptstyle L'G'h}\downarrow \\ L'G'c & \xrightarrow{Go} & Gy & \xleftarrow{Gp} & L'G'd \end{array}$$

and

$$\begin{array}{ccccc} L'a' & \xrightarrow{m'} & x' & \xleftarrow{n'} & L'b' \\ {\scriptstyle L'f'}\downarrow & & {\scriptstyle g'}\downarrow & & {\scriptstyle L'h'}\downarrow \\ L'c' & \xrightarrow{o'} & y' & \xleftarrow{p'} & L'd' \end{array} \quad \xmapsto{\Theta'} \quad \begin{array}{ccccc} LF'a' & \xrightarrow{Fm'} & Fx' & \xleftarrow{Fn'} & LF'b' \\ {\scriptstyle LF'f'}\downarrow & & {\scriptstyle Fg'}\downarrow & & {\scriptstyle LF'h'}\downarrow \\ LF'c' & \xrightarrow{Fo'} & Fy' & \xleftarrow{Fp'} & LF'd' \end{array}$$

Proof. In light of Theorem 3.5, it suffices to show that $\Theta \dashv \Theta'$ gives a geometric morphism.

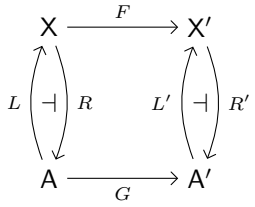
Let ℓ denote the L-structured cospans $La \to x \leftarrow Lb$ and ℓ' denote the L'-structured cospan $L'a' \to x' \leftarrow L'b'$. Denote the unit and counit for $F \dashv G$ by η, ε and for $F' \dashv G'$ by η', ε'. The assignments

$$((f,g,h)\colon \ell \to \Theta'\ell') \mapsto ((\varepsilon' \circ F'f, \varepsilon \circ Fg, \varepsilon' \circ F'h)\colon \Theta\ell \to \ell')$$
$$((f',g',h')\colon \Theta\ell \to \ell') \mapsto ((G'f' \circ \eta', Gg' \circ \eta, G'h' \circ \eta')\colon \ell \to \Theta'\ell')$$

give a bijection $\hom(\Theta\ell, \ell') \simeq \hom(\ell, \Theta'\ell')$. Moreover, it is natural in ℓ and ℓ'. This rests on the natural maps η, ε, η', and ε'. The left adjoint Θ' preserves finite limits because they are taken pointwise and L, F, and F' all preserve finite limits. □

We end this section by organizing the categories of the form ${}_L\mathsf{StrCsp}$ into a 2-category.

Definition 3.7. Let $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ and $L' \dashv R' \colon \mathsf{X}' \to \mathsf{A}'$ be geometric morphisms. A **morphsim of structured cospan categories** ${}_L\mathsf{StrCsp} \to_{L'} \mathsf{StrCsp}$ is pair of finitely continuous and cocontinuous functors $F \colon \mathsf{X} \to \mathsf{X}'$ and $G \colon \mathsf{A} \to \mathsf{A}'$ fitting into the commuting diagram



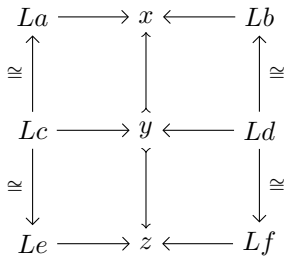
The reader may check that a morphism of structured cospan categories gives a functor from ${}_L\mathsf{StrCsp}$ to ${}_{L'}\mathsf{StrCsp}$.

Structured cospan categories fit as objects into a 2-category **StrCsp**. The 1-arrows are their morphisms and 2-arrows of type $(F, G) \Rightarrow (F', G')$ are pairs of natural transformations $\alpha \colon F \Rightarrow F'$ and $\beta \colon G \to G'$.

4. Rewriting structured cospans

We now know that the category ${}_L\mathsf{StrCsp}$ of structured cospans and their morphisms is a topos and, therefore, support a rich rewriting theory. In this section, we develop this theory.

Definition 4.1. A **rewrite rule of structured cospans** is an isomorphism class of spans of structured cospans of the form



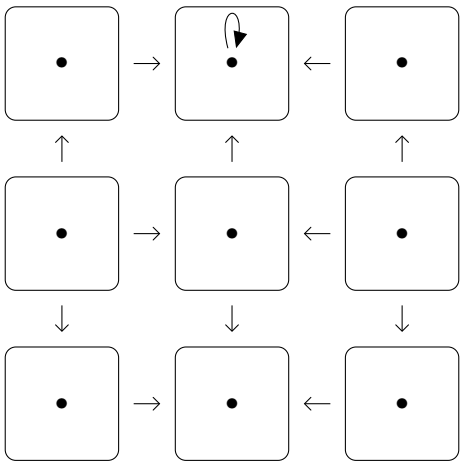
The marked arrows $\rightarrowtail$ are monic.

The conceit of this rule is that the structured cospan in the top row of the diagram replaces the structured cospan in the bottom row. This is similar to a typical rewrite rule except that we are now orienting our diagrams as "top-replaces-bottom" instead of "left-replaces-right". A non-superficial difference is that this diagram is not merely a span in ${}_L\mathsf{StrCsp}$ with monic legs. We force the spans between the inputs and outputs to have invertible legs, effectively preventing the interfaces from changing in a rewrite. This constraint is required for the double category ${}_L\mathbb{S}\mathbf{trCsp}$ in Definition 4.4 to satisfy the interchange law. In less technical terms, the constraint ensures that rewriting interacts nicely with structured cospan composition.

Example 4.2. We illustrate this by returning to our running example of modelling the internet. This time, instead of modeling the internet with a graph, we use an

open graph as defined via structured cospans in Example 3.3. To remove loops in our model, we can introduce the rule

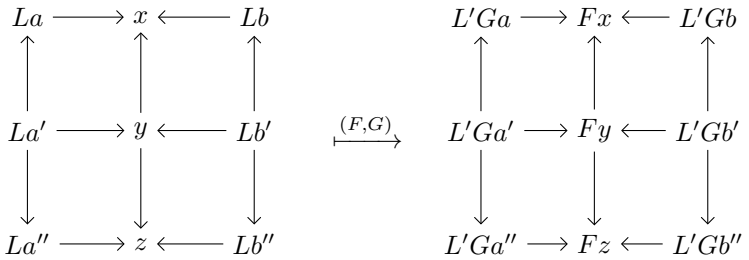
(7)



which removes a loop on any node that is both an input and output.

Before we see how to apply this rule, however, we need to further develop our theory. Next, we look at grammars and derived rules for structured cospans.

4.1. **Structured cospan grammars.** In Definition 2.1, we defined a category Gram of grammars and their morphisms. Our interest now shifts to the subcategory of Gram spanned by the structured cospan grammars. By a **structured cospan grammar**, we mean a structured cospan category paired with a set of rewrite rules of structured cospans. A morphism a structured cospan grammars $(F, G)\colon ({}_L\mathsf{StrCsp}, P) \to ({}_{L'}\mathsf{StrCsp}, P')$ is a morphism of structured cospan categories (Definition 3.7) with the property that P' contains the image of each rule in P. Note that we defined a morphism of structured cospan categories in a way so that

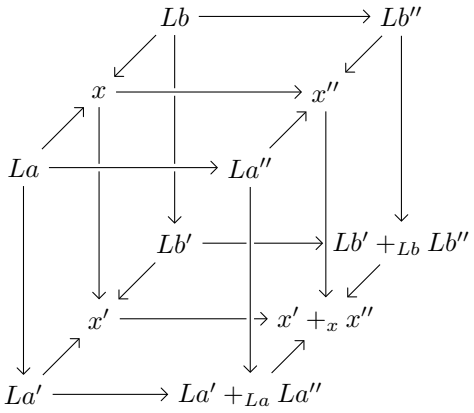


thus ensure that the image of a rule has a form suitable for P'.

Ultimately, we want to associate a rewrite relation to each structured cospan grammar. To do this, we use the notion of a derived rule. But we do need to be careful to ensure that, in the context of structured cospans, derived rules have the proper form.

4.2. **Derived rules are rules.** Derived rules emerge from pushouts, which in the category ${}_L\mathsf{StrCsp}$ have the form

(8)

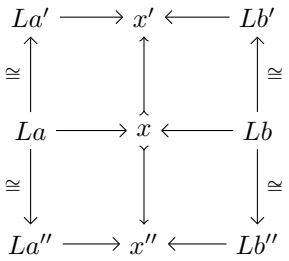


The legs of the span

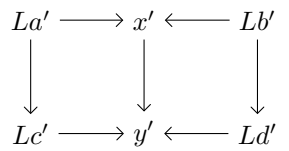
$$La' +_{La} La'' \leftarrow x' +_x x'' \rightarrow Lb' +_{Lb} Lb''$$

are induced by the universal property of the pushouts $La' +_{La} La''$ and $Lb' +_{La} Lb''$. This same universal property ensures that the bottom and right faces of Diagram (8) commute. Moreover, this span *is* an L-structured cospan because as a left adjoint L preserves pushouts so is isomorphic to $L(a' +_a a'') \leftarrow x' +_x x'' \rightarrow L(b' +_b b'')$.

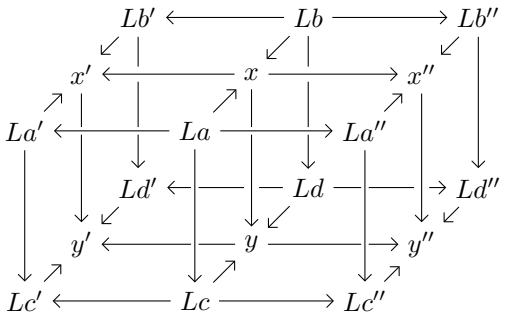
Consider a structured cospan rule



and morphism



that has a pushout complement. This data induces a derived rule that appears on the bottom face of the diagram



This derived rule is still a rule because pushouts in a topos preserve monos and epis which, because topoi are balanced, implies that pushouts also preserve isomorphisms. In short, derived rules are rules.

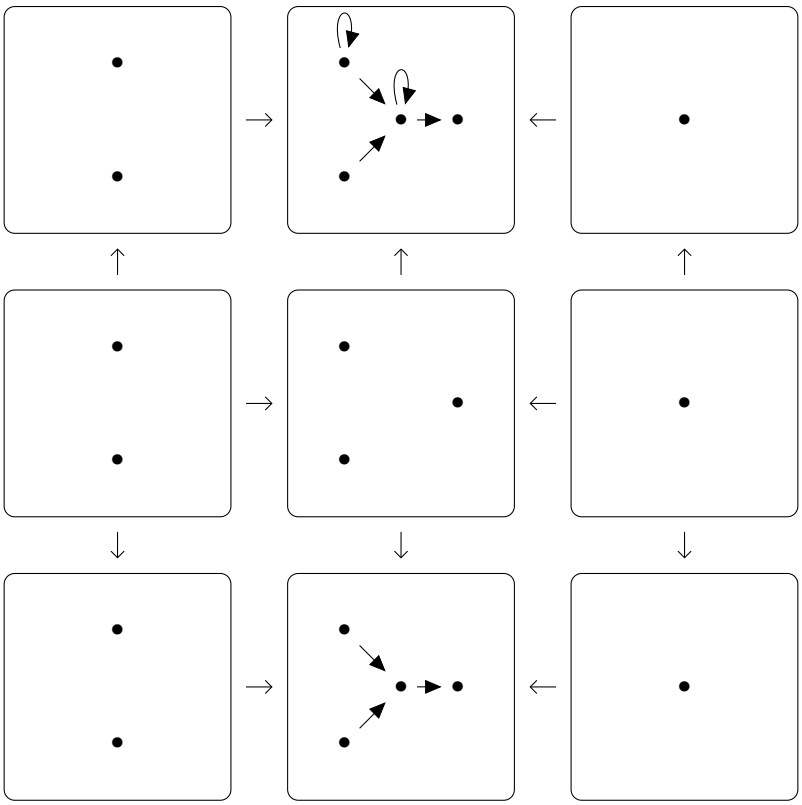
4.3. **Rewrite relation.** Now that we are certain that derivation preserves rewrite rules of structured cospans, we can look at the rewrite relation. We can try to encode the rewrite relation as arrows in a category as we did with the lanugage functor $\mathsf{Lang}\colon \mathsf{Gram} \to \mathsf{Cat}$ in Section 2, however, this would fail to capture the added layer of composition coming from the structured cospans. Instead, we give the type $\mathsf{Lang}\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$ to the language functor for structured cospans. The codomain, here, is the category of double categories and their functors. The language $\mathsf{Lang}({}_L\mathsf{StrCsp}, P)$, then, is a double category with structured cospans as horizontal arrows and certain rewrite rules as squares. Before defining Lang precisely, we sketch a simple example to help visualize the language for a structured cospan grammar.

Example 4.3. Starting with the discrete graph geometric morphism,

$$\mathsf{RGraph} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{Set}$$

for which ${}_L\mathsf{StrCsp}$ is the category of open graphs, consider a grammar P comprising only Rule 7. Then $\mathsf{Lang}({}_L\mathsf{StrCsp}, P)$ is the double category whose horizontal arrows are open graphs and the existence of a square means that the bottom open graph can be obtained by removing loops from the top. For instance, $\mathsf{Lang}({}_L\mathsf{StrCsp}, P)$

contains the square



To give a rigorous definition of Lang, we first need to define the double category in which $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ is generated.

Definition 4.4. Fix a geometric morphism $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. There is a double category ${}_L\mathbb{S}\mathbf{trCsp}$ whose objects are those from A, whose vertical arrows are spans with invertible legs in A, whose horiztonal arrows are L-structured cospans, and whose squares are L-structured cospan rewrite rules.

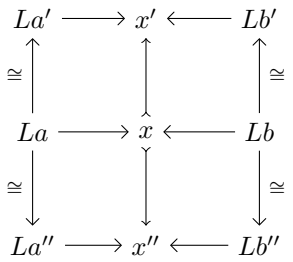
Showing that ${}_L\mathbb{S}\mathbf{trCsp}$ is a double category largely involves checking the requisit list of axioms. The non-trivial part of this is showing the interchange law which has been shown in previous work [4, Lem. 4.2, Lem. 4.3]. We should note that ${}_L\mathbb{S}\mathbf{trCsp}$ is actually a symmetric monoidal double category via pointwise addition [4, Lem. 4.4] (this uses the fact that L preserves coproducts), however, this structure plays no role in this work, so we do not mention it again.

The following theorem gives the language functor construction.

Theorem 4.5. *Let $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ and $L' \dashv R' \colon \mathsf{X}' \to \mathsf{A}'$ be geometric morphisms. Let $({}_L\mathsf{StrCsp}, P)$ and $({}_{L'}\mathsf{StrCsp}, P')$ be structured cospan grammars. Let dP be the set of all structured cospan rewrite rules derived from the rules in P. Define a relation $\rightsquigarrow$ on L-structured cospans by setting*

$$(La' \to x' \leftarrow Lb') \rightsquigarrow (La'' \to x'' \leftarrow Lb'')$$

if and only if dP contains a rule



There exists a double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ *whose objects are those of* A*, vertical arrows are spans with inverible legs in* A*, horizontal arrows are* L*-structured cospans, and squares are generated by the rules in* dP*.*

Given a morphism of structured cospan grammars

$$(F, G) \colon ({}_L\mathsf{StrCsp}, P) \to ({}'_L\mathsf{StrCsp}, P'),$$

there is a morphism of double categories

$$\mathrm{Lang}(F, G) \colon \mathrm{Lang}({}_L\mathsf{StrCsp}, P) \to \mathrm{Lang}({}'_L\mathsf{StrCsp}, P)$$

defined by $a \mapsto Fa$ *on objects and extended from* $(x \rightsquigarrow y) \mapsto (Fx \rightsquigarrow Fy)$ *on arrows.*

We have, thus, defined a functor $\mathrm{Lang} \colon \mathsf{StrCspGram} \to \mathsf{DblCat}$*.*

Proof. Define $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ to be the sub-double-category of ${}_L\mathbb{S}\mathbf{trCsp}$ generated by the squares in dP.

Given a morphism of structured cospan grammars

$$(F, G) \colon ({}_L\mathsf{StrCsp}, P) \to ({}_{L'}\mathsf{StrCsp}, P'),$$

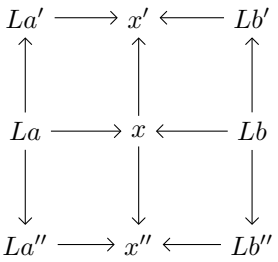
we can define a double functor by extending the definition of F on the generating squares. It is certain that $\mathrm{Lang}(F, G)$ sends a generating square to a generating square because F preserves rules and, because F is finitely cocontinuous, preserves derived rules too. Hence $\mathrm{Lang}(F, G)$ sends a generating square in $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ to a generating square in $\mathrm{Lang}({}_{L'}\mathsf{StrCsp}, P')$. □

By construction, Lang soundly encodes the rewrite relation into the squares of a double category.

Corollary 4.6. *Let* $\rightsquigarrow^*$ *be the reflexive and transitive closure of* $\rightsquigarrow$*. If*

$$(La' \leftarrow x' \rightarrow Lb') \rightsquigarrow^* (La'' \leftarrow x'' \rightarrow Lb''),$$

then there is a square



in $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$

When rewriting in topoi, we witnessed an equivalence between the rewrite relation and language (Theorem 2.2). When rewriting structured cospans, however, we do not expect such an equivalence for structured cospans because the rewrite relation $\rightsquigarrow^*$ only encompasses the compositionality of the rules, whereas the language $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ captures the compositionality of both the rules and structured cospans. Because of this additional compositional structure, we argue that the rewrite relation is not the morally correct semantics to study in the case of structured cospans.

5. Expressiveness of grammars

In general, for any rewrite rule $\ell \leftarrow k \rightarrow r$, there is only one constraint on the value of k: it must be a subobject of ℓ and r. But requiring that k also be *discrete* can simplify any anaylsis involving that rule. This leads us question whether we can learn about a grammar (T, P) by instead studying the grammar $(\mathsf{T}, \flat P)$, where $\flat P$ is obtained by discretizing the apexes of every rule in P. In this section we explain the $\flat$ notation and make precise the concept of discreteness before giving the main result of this section which states that (T, P) and $(\mathsf{T}, P_\flat)$ have the same rewrite relation under certain conditions. This result generalizes a characterization of discrete graph grammars given by Ehrig, et. al. [5, Prop. 3.3].

Experts in topos theory know that discreteness comes from the flat modality on a local topos. However, we avoid the lengthy detour required to unpack the meaning of "the flat modality on a local topos" because it does not benefit our story (curious readers can find this information elsewhere [7, Ch. 3.6]). Instead, we offer a more direct, if less subtle, approach by defining a *discrete comonad* in a way to fit our needs.

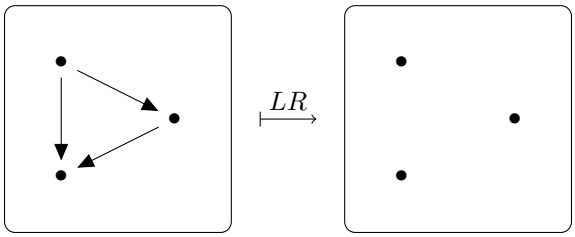
Definition 5.1 (Discrete comonad)**.** A comonad on a topos is called **discrete** if its counit is monic. We use $\flat$ to denote a discrete comonad.

We can interpret a discrete comonad as returning largest interface $\flat x$ supported by a system x. If x is a graph, then $\flat x$ is the discrete graph underlying x. Here is a brief example illutrating how the discrete graph adjunction we have made such use of gives rise to $\flat$.

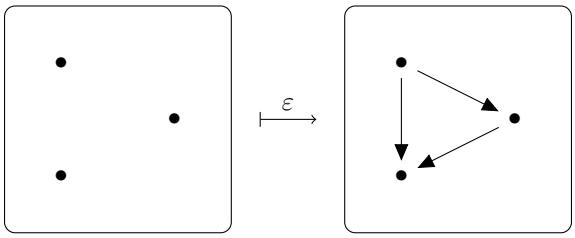
Example 5.2. Consider the geometric morphism

$$\mathsf{Graph} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{Set}$$

defined by setting La to be the discrete graph on a and Rx to be the set of nodes in x. This adjunction induces the comonad $\flat := LR$ on Graph. Applying LR to a graph x returns the discrete graph underlying x, for instance



The counit $\varepsilon_x \colon LRx \to x$ is certainly monic as it includes the discrete graph LRx into the graph x, as in



Discrete comonads provide a tool to control the form of a grammar by replacing every rule $\ell \leftarrow k \to r$ with $\ell \leftarrow \flat k \to r$.

Definition 5.3 (Discrete grammar)**.** Let $\flat \colon \mathsf{T} to \mathsf{T}$ be a discrete comonad with counit ε. Given a grammar (T, P), define $P_\flat$ as the set containing

$$\ell \leftarrow k \xleftarrow{\varepsilon} \flat k \xrightarrow{\varepsilon} k \to r$$

for each rule $\ell \leftarrow k \to r$ in P. We call $(\mathsf{T}, P_\flat)$ the **discrete grammar** underlying (T, P).

The main result of this section, Theorem 5.6, says that the grammars (T, P) and $(\mathsf{T}, P_\flat)$ have the same rewrite relation when, for each rule $\ell \leftarrow k \to r$ in P, the subobject lattice $\mathrm{Sub}(k)$ has all meets. A reasonable concern is that requiring $\mathrm{Sub}(k)$ to have all meets is overly restrictive but, in fact, a fairly large class of grammars has this property. For instance, this class includes any grammar built on a presheaf category. This is especially pertinent because many systems can be formalized using labelled graphs, which are presheaves. The ZX-calculus [3] serves as one example.

Proposition 5.4. *Fix an object k of a topos T. The subobject lattice $\mathrm{Sub}(k)$ has arbitrary meets when the over-category $T \downarrow k$ has either all products or all coproducts.*

Proof. Because $T \downarrow t$ is a topos, it has equalizers. Thus giving it all products ensures the existence of all limits, hence meets.

In general, a lattice with arbitrary joins also has arbitrary meets: define the meet of a subset to be the join of all its lower bounds. Because any join in $\mathrm{Sub}(k)$ is a coproduct in $T \downarrow k$, assuming all coproducts provides all joins and, therefore, all meets. □

Corollary 5.5. *Any suboject lattice in a presheaf categories has arbitrary meets.*

Proof. An over-category of presheaves is again a presheaf category, hence has all coproducts. □

At last, we reach our main result of this section. This result mirrors Chomsky's heirarchy of grammars [2] and the classification of graph grammars [5, Prop. 3.3].

Theorem 5.6. *Let T be a topos, $\flat \colon \mathsf{T} \to \mathsf{T}$ be a discrete comonad, and (T, P) be a grammar such that, for every rule $\ell \leftarrow k \to r$ in P, the subobject lattice $\mathrm{Sub}(k)$ has all meets. The rewrite relation for (T, P) equals the rewrite relation for its underlying discrete grammar $(\mathsf{T}, P_\flat)$.*

Proof. Suppose that (T, P) induces $g \rightsquigarrow h$. That means there exists a rule $\ell \leftarrow k \rightarrow r$ in P and a derivation

(9)
$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \downarrow & & \downarrow & & \downarrow \\ g & \longleftarrow & d & \longrightarrow & h \end{array}$$

We can achieve that same derivation using rules in $P_\flat$, which requires that we build a pushout complement w of the diagram

$$\begin{array}{ccc} k & \xleftarrow{\varepsilon} & \flat k \\ \downarrow & & \\ d & & \end{array}$$

Because $\mathrm{Sub}(k)$ has all meets, we can define $w \coloneqq \bigwedge\{z \colon z \vee k = d\} \vee \flat k$ which comes with inclusions $\flat k \rightarrow w$ and $w \rightarrow d$. Note that $w \vee k = d$ and $w \wedge k = \flat k$. It follows that

$$\begin{array}{ccc} k & \longleftarrow & \flat k \\ \downarrow & & \downarrow \\ d & \longleftarrow & w \end{array}$$

is a pushout with which we get a derivation

(10)
$$\begin{array}{ccccccccc} \ell & \longleftarrow & k & \longleftarrow & \flat k & \longrightarrow & k & \longrightarrow & r \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ g & \longleftarrow & d & \longleftarrow & w & \longrightarrow & d & \longrightarrow & h \end{array}$$

with respect to $P_\flat$. Therefore, $g \rightsquigarrow h$ via P in Diagram (9) implies that $g \rightsquigarrow^* h$ via $P_\flat$ as shown in Diagram (10).

For the other direction, suppose $P_\flat$ induces $g \rightsquigarrow h$ via a derivation

(11)
$$\begin{array}{ccccc} \ell & \longleftarrow & \flat k & \longrightarrow & r \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle \theta} & & \downarrow{\scriptstyle m'} \\ g & \underset{\psi}{\longleftarrow} & d & \longrightarrow & h \end{array}$$

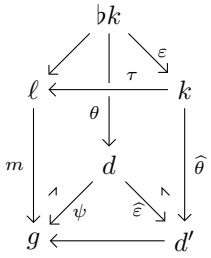
By construction, the rule $\ell \leftarrow \flat k \rightarrow r$ in $P_\flat$ was induced by some rule

(12)
$$\ell \xleftarrow{\tau} k \rightarrow r$$

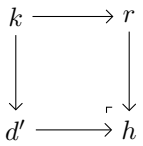
in P. Define d' to be the pushout of the diagram

$$\begin{array}{ccc} \flat k & \xrightarrow{\varepsilon} & k \\ {\scriptstyle \theta}\downarrow & & \downarrow{\scriptstyle \widehat{\theta}} \\ d & \underset{\widehat{\varepsilon}}{\longrightarrow} & d' \end{array}$$

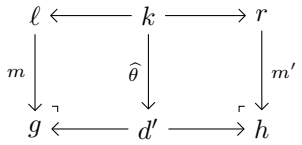
Given the maps $\psi\colon d \to g$ and $m\tau\colon k \to \ell \to g$ from (11) and (12), we invoke the universial property of d' to get a canonical map $d' \to g$. This map fits into the diagram



whose back faces are pushouts. Using a standard diagram chasing argument, we can show that the front face is also a pushout. Similarly, the square



is a pushout. Sticking these two pushouts together gives the double pushout diagram



proving that P induces $g \rightsquigarrow h$. □

6. An inductive view of rewriting in a topos

Before graph rewriting, there was formal language rewriting and term rewriting. In these cases, there are two ways to define the rewrite relation. The first way is called the *operational method*, which applies a rule by substituting a sub-term for another term. The second way is called the *inductive method*, which constructs the rewrite relation using generators and closure operations. In classical graph rewriting, only the operational method existed, where substitution was achieved with the double pushout method. Eventually, Gadducci and Heckel introduced an inductive method to construct the rewrite relation, opening the way to analyze graph grammars through structural induction. In this section, we adapt their ideas in order to give an inductive definition of the rewrite relation for a grammar (X, P) where X is a topos subject to several conditions. In the systems interpretation, this section is about generating the rewrite relation of closed systems using a generating set of open systems.

We start by formalizing a closed system as structured cospans with an empty interface $L0 \to x \leftarrow L0$. To decompose a closed system $L0 \to x \leftarrow L0$ is to write it

as a composite

$$L0 \to x_1 \leftarrow La_1 \to x_2 \leftarrow La_2 \to \cdots \leftarrow La_{n-1} \to x_n \leftarrow L0$$

We use such decompositions to prove our main result which states that two closed systems are equivalent precisely when, in the language double category, there is a square between them. This result shows two things. First, constructing the rewrite relation for a closed system is functorial. Second, it justifies the idea that open systems provide a local perspective on closed systems via this decomposition.

To decompose closed systems, considered as objects of a topos X into open systems, we need a grammar (X, P). We also need to equip closed systems with interfaces, which we do using an adjunction

$$\mathsf{X} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{A}$$

where L preserves pullbacks and has a monic counit. This adjunction induces a discrete comonad $\flat := LR$ so we can form the discrete grammar $(\mathsf{X}, P_\flat)$. Now define the structured cospan grammar $({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ where $\widehat{P_\flat}$ contains the rule

(13)
$$\begin{array}{ccccc}
L0 & \longrightarrow & \ell & \longleftarrow & LRk \\
\uparrow & & \uparrow & & \uparrow \\
L0 & \longrightarrow & LRk & \longleftarrow & LRk \\
\downarrow & & \downarrow & & \downarrow \\
L0 & \longrightarrow & r & \longleftarrow & LRk
\end{array}$$

for each rule $\ell \leftarrow LRk \to r$ of $P_\flat$. This grammar plays an important role in proving our main theorem. Also needed for that proof is the following lemma that formalizes the analogy between rewriting the disjoint union of systems and tensoring squares.

Lemma 6.1. *If $x \rightsquigarrow^* y$ and $x' \rightsquigarrow^* y'$, then $x + x' \rightsquigarrow^* y + y'$*

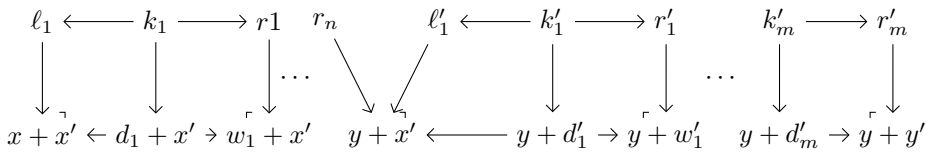
Proof. If the derivation $x \rightsquigarrow^* y$ comes from a string of double pushout diagrams

$$\begin{array}{ccccccccccc}
\ell_1 & \longleftarrow & k_1 & \longrightarrow & r_1 & & \ell_2 & \longleftarrow & k_2 & \longrightarrow & r_2 \quad k_n \longrightarrow r_n \\
\downarrow & & \downarrow & & \searrow & & \swarrow & & \downarrow & & \downarrow \quad \cdots \quad \downarrow \\
x & \longleftarrow & d_1 & & \longrightarrow & & w_1 & \longleftarrow & d_2 & \longrightarrow & w_2 \quad d_n \longrightarrow y
\end{array}$$

and the derivation $x' \rightsquigarrow^* y'$ comes from a string of double pushout diagrams

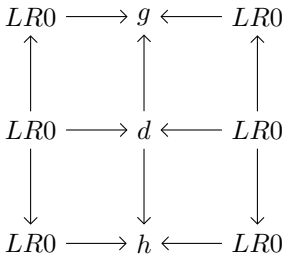
$$\begin{array}{ccccccccccc}
\ell'_1 & \longleftarrow & k'_1 & \longrightarrow & r'_1 & & \ell'_2 & \longleftarrow & k'_2 & \longrightarrow & r'_2 \quad k'_m \longrightarrow r'_m \\
\downarrow & & \downarrow & & \searrow & & \swarrow & & \downarrow & & \downarrow \quad \cdots \quad \downarrow \\
x' & \longleftarrow & d'_1 & & \longrightarrow & & w'_1 & \longleftarrow & d'_2 & \longrightarrow & w'_2 \quad d'_m \longrightarrow y'
\end{array}$$

realize $x + x' \rightsquigarrow^* y + y'$ by the diagram



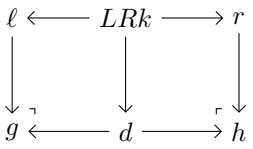
□

Theorem 6.2. *Fix an adjunction* $(L \dashv R)\colon \mathsf{X} \rightleftarrows \mathsf{A}$ *with monic counit. Let* (X, P) *be a grammar such that for every* X*-object* x *in the apex of a production of* P*, the lattice* $\mathrm{Sub}(x)$ *has all meets. Given* g*,* $h \in \mathsf{X}$*, then* $g \rightsquigarrow^* h$ *in the rewrite relation for a grammar* (X, P) *if and only if there is a square*

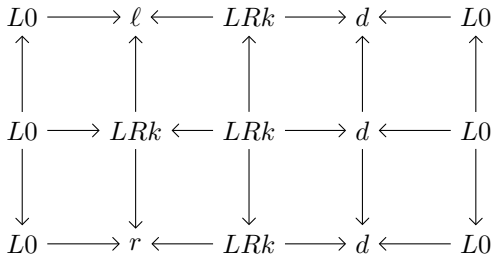


in the double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$.

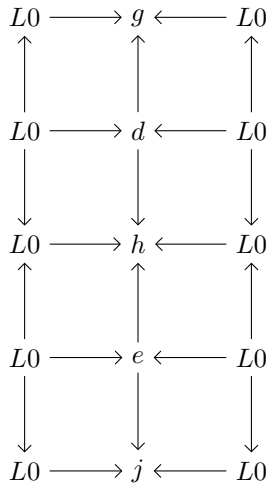
Proof. We show sufficiency by inducting on the length of the derivation. If $g \rightsquigarrow^* h$ in a single step, meaning that there is a diagram



then the desired square is the horizontal composition of

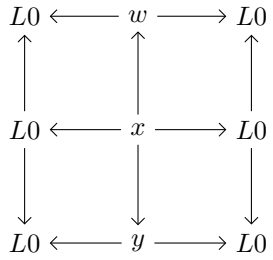


The left square is a generator and the right square is the identity on the horizontal arrow $LRk \to d \leftarrow L0$. The square for a derivation $g \rightsquigarrow^* h \rightsquigarrow j$ is the vertical composition of

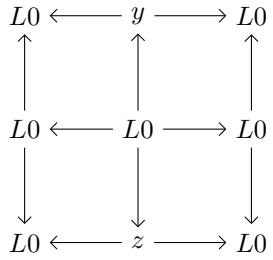


The top square is from $g \rightsquigarrow^* h$ and the second from $h \rightsquigarrow j$.

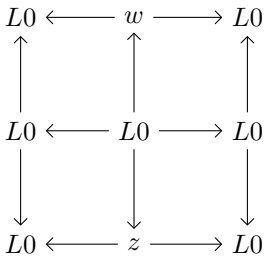
Conversely, proceed by structural induction on the generating squares of $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$. It suffices to show that the rewrite relation is preserved by vertical and horizontal composition by generating squares. Suppose we have a square



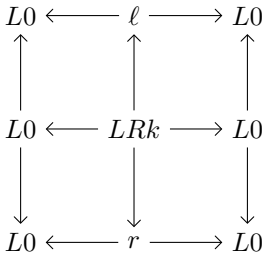
corresponding to a derivation $w \rightsquigarrow^* y$. Composing this vertically with a generating square, which must have form



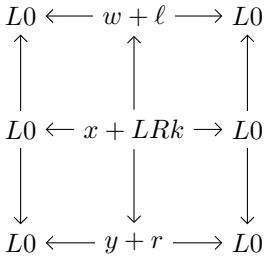
corresponding to a rule $y \leftarrow L0 \rightarrow z$ gives



which corresponds to a derivation $w \rightsquigarrow^* y \rightsquigarrow z$. Composing horizontally with a generating square

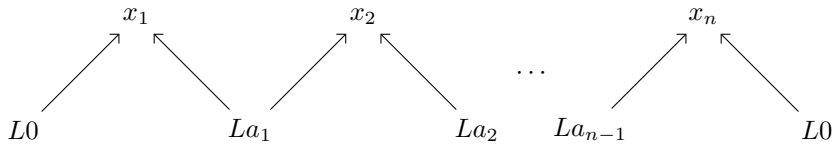


corresponding with a rule $\ell \leftarrow LRk \rightarrow r$ results in the square

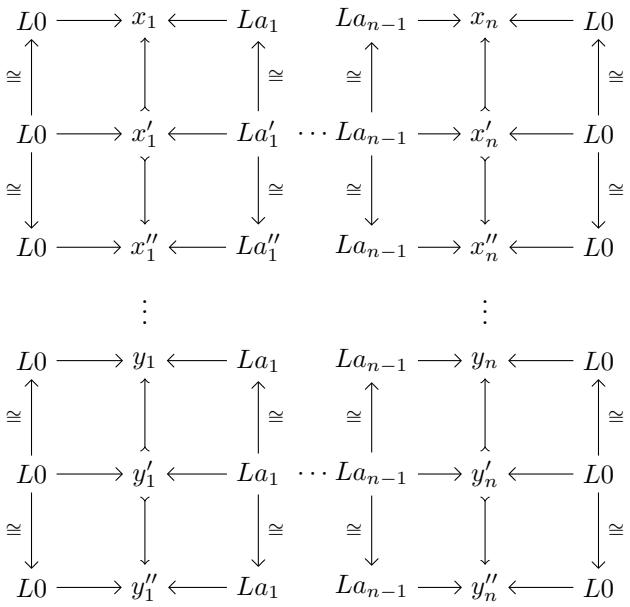


But $w + \ell \rightsquigarrow^* y + r$ as seen in Lemma 6.1. □

With this result, we have completely described the rewrite relation for a grammar (X, P) with squares in $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ framed by the initial object of X. These squares are rewrites of a closed system in the sense that the interface is empty. We can instead begin with a closed system x in X as represented by a horizontal arrow $L0 \rightarrow x \leftarrow L0$ in $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ and decompose it into a composite of sub-systems, that is a sequence of composable horizontal arrows



Rewriting can be performed on each of these sub-systems



The composite of these squares is a rewriting of the original system.

References

[1] John Baez and Kenny Courser. Structured cospans. *In Preperation.* (Referred to on page 3, 11.)

[2] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959. (Referred to on page 22.)

[3] Daniel Cicala. Categorifying the zx-calculus. In Bob Coecke and Aleks Kissinger, editors, Proceedings 14th International Conference on *Quantum Physics and Logic,* Nijmegen, The Netherlands, 3-7 July 2017, volume 266 of *Electronic Proceedings in Theoretical Computer Science*, pages 294–314. Open Publishing Association, 2018. (Referred to on page 4, 22.)

[4] Daniel Cicala and Kenny Courser. Spans of cospans in a topos. *Theory Appl. Categ.*, 33:Paper No. 1, 1–22, 2018. (Referred to on page 19.)

[5] Harmut Ehrig, Michael Pfender, and Hans Schneider. Graph-grammars: an algebraic approach. In *14th Annual IEEE Symposium on Switching and Automata Theory (Univ. Iowa, Iowa City, Iowa, 1973)*, pages 167–180. 1973. (Referred to on page 8, 9, 21, 22.)

[6] Fabio Gadducci and Reiko Heckel. An inductive view of graph transformation. In *Recent trends in algebraic development techniques (Tarquinia, 1997)*, volume 1376 of *Lecture Notes in Comput. Sci.*, pages 223–237. Springer, Berlin, 1998. (Referred to on page 6.)

[7] Peter Johnstone. *Sketches of an elephant: A topos theory compendium*, volume 2. Oxford University Press, 2002. (Referred to on page 21.)

[8] Stephen Lack and PawełSobociński. Adhesive categories. In *Foundations of software science and computation structures*, volume 2987 of *Lecture Notes in Comput. Sci.*, pages 273–288. Springer, Berlin, 2004. (Referred to on page 3, 9, 10.)

[9] Stephen Lack and PawełSobociński. Toposes are adhesive. In *Graph transformations*, volume 4178 of *Lecture Notes in Comput. Sci.*, pages 184–198. Springer, Berlin, 2006. (Referred to on page 3, 9.)

[10] Gavin Wraith. Artin glueing. *J. Pure Appl. Algebra*, 4:345–348, 1974. (Referred to on page 14.)

Email address: dcicala@newhaven.edu

Department of Mathematics and Physics, University of New Haven