REWRITING STRUCTURED COSPANS

DANIEL CICALA

Abstract. To foster the study of networks on an abstract level, we further study the formalism of *structured cospans* introduced by Baez and Courser. A structured cospan is a diagram of the form $La \to x \leftarrow Lb$ built from a geometric morphism with left exact left adjoint $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. We show that this construction is functorial and results in a topos with structured cospans for objects. Additionally, structured cospans themselves are compositional. Combining these two perspectives, we define a double category of structured cospans. We then leverage adhesive categories to create a theory of rewriting for structured cospans. We generalize the result from graph rewriting stating that a graph grammar induces the same rewrite relation as its underlying graph grammar. We use this fact to prove our main result, a complete characterization of the rewriting relation for a topos X using double categories. This provides a compositional framework for rewriting systems.

1. Introduction

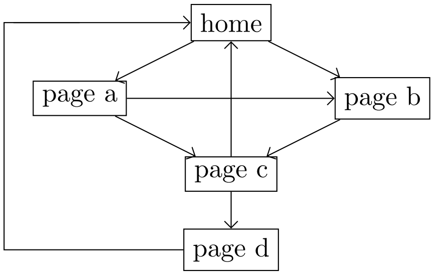
Structured cospans are a framework for reasoning about systems with inputs and outputs. Rewriting is a topic that covers methods to replace parts of an object with another part. Perhaps the two most deeply explored objects are words on a given alphabet and graphs. In this paper, we introduce rewriting to structured cospans.

While the term “system” is intentionally vague, we have in mind those that can be described as a collection of discrete objects that are somehow related. Examples

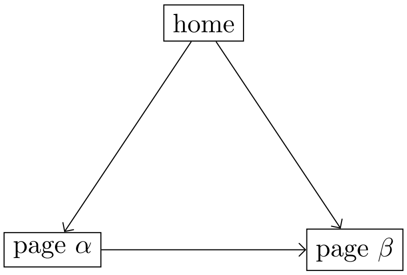
Date: January 6, 2020.

include electrical systems in a house consisting of appliances connected by wires; social systems where people are connected by social relationships; virtual systems such as a collection of webpages connected by links. Often, one will analyze a fixed, closed system and ignore the capability for compatible systems to be connected to form larger systems.

Baez and Courser introduced structured cospans as an abstract framework to model such systems [1]. For example, suppose we want to analyze the structure of Professor Smith's academic's website, which contains two sections. The section devoted to research comprises, say, five pages connected by several links and can be modelled as the graph



. The section devoted to research comprises, say, three pages connected by several links and can be modelled as the graph



.

(Include something like: "We move through three levels of abstraction. The most abstract involves working with topoi. The intermediate level involves working with gerenic systems that can connect together when compatible. At our least abstract, we will work with a particular system, a running example will be modeling the internet with a graph.")

(Should I write at the level of systems in all exposition and swtich to topos only if absolutely needed?)

(Should I replace all of my graph examples with nodes spelling actual websites, like `google` with a little box around them or something? Or maybe model a person website with pages like `checkout`, or `products` and `home` ect.)

1.1. Required background.

(Say something about not needing topos theory strictly, it can be black boxed or it suffice to consider the particular example of graphs.)

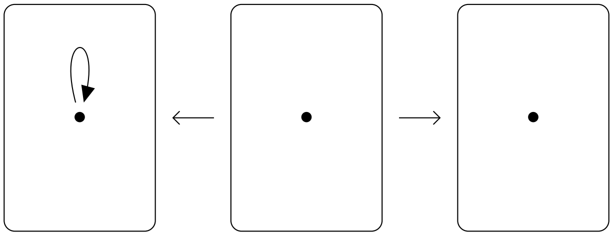
2. Rewriting in topoi

A common tool to model a system of objects and relations 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 wired in series and a single resistor.

Double pushout rewriting has an established literature, so we point the interested reader to Lack and Sobocinski [8] for an in-depth treatment. Here, we merely cover the fundamentals and do so primarily to establish our conventions. Lack and Sobocinski axiomatize rewriting using what they call *adhesive categories*. We do not use the full generality of the class of adhesive categories and, instead, restrict our attention to the sub-class of topoi [9]. Indeed, everything that follows occurs in 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



(1)

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 $\mathsf{2} + \mathsf{2}$ with the string $\mathsf{4}$. We capture 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 $(\mathsf{S}, P) \to (\mathsf{T}, Q)$ is a pullback and pushout preserving functor $F \colon \mathsf{S} \to \mathsf{T}$ 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 (1). 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 $m \colon \ell \to \ell'$ for which there exists a pushout complement, that is an object k' fitting into a pushout diagram

$$\begin{tikzcd}
\ell \arrow[d, "m"'] & k \arrow[l] \arrow[d] \\
\ell' \arrow[ur, phantom, "\ulcorner", very near start] & k' \arrow[l]
\end{tikzcd}$$

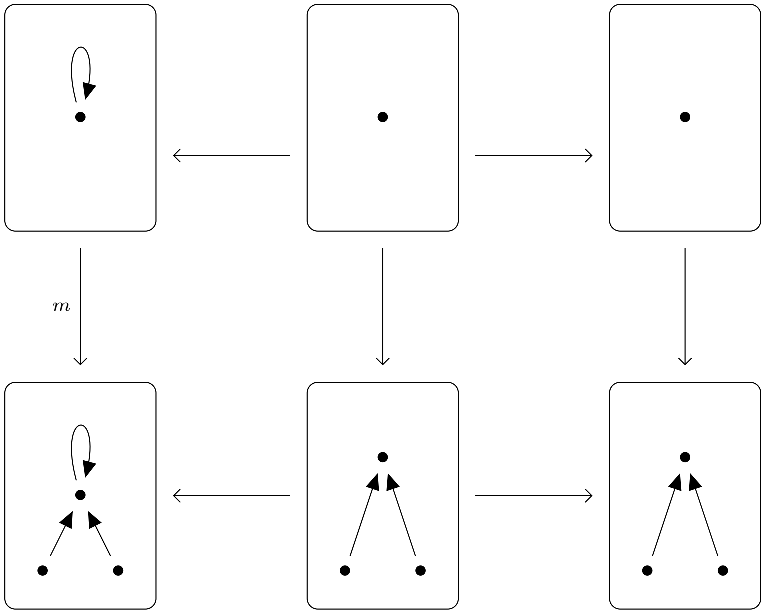
A pushout complement need not exist, but if it does and the map $k \to \ell$ is monic, then it is unique up to isomorphism [8, Lem. 15].

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

$$\begin{tikzcd}
\ell \arrow[d, "m"'] & k \arrow[l] \arrow[r] \arrow[d] & r \arrow[d] \\
\ell' \arrow[ur, phantom, "\ulcorner", very near start] & k' \arrow[l] \arrow[r] & r' \arrow[ul, phantom, "\urcorner", very near start]
\end{tikzcd}$$

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.

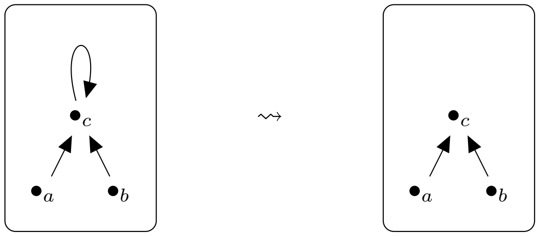
In our model of the internet, we can apply Rule (1) to any node with a loop. The double pushout diagram



(2)

illustrates one possible application and gives, in its bottom row, a derived rule that can be further applied to any suitable graph.

A grammar (T, P) induces a collection dP containing any derived rule 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* $\rightsquigarrow^*$. Roughly, $x \rightsquigarrow^* y$ if we can rewrite x into y by apply a sequence of rules in dP. To 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 $\ell' \leftarrow k' \rightarrow r'$ in dP. For instance, Diagram (2) 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. In deference to the linguistic genesis of rewriting theory, we denote this functor Lang, short for "language".

Theorem 2.2. *Given a grammar (T, P), 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 (T, P) be a grammar and x and y be objects of T. There is an arrow $x \to y$ in $\mathrm{Lang}(\mathsf{T}, P)$ if and only if $x \rightsquigarrow^* y$ by the rewrite relation.*

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. □

While this section leaves the practice of rewriting to other sources [5], we have developed enough to blend the theory of rewriting with structured cospans.

3. Structured cospans form a topos

Having reviewed rewriting, we can now begin our story proper. In this section, we reintroduce the notion of structured cospans and then show that, under mild assumptions, they form a topos.

Baez and Courser [1] introduced structured cospans to serve as a syntax for open systems. A system x is **open** when equipped with a mechanism by which it can connect to a compatible system y. This "connection operation" forms a new system: the union of x and y. 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, such as an isolated pendulum or electrical circuit.

In order to rewrite structured cospans in the sense of Section 2, they must form a topos. Therefore, we impose stronger conditions than required by Baez and Courser. To specify these conditions, we use "geometric morphisms" from topos theory. A **geometric morphism** $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ between topoi X and A is an adjunction

$$\begin{tikzcd} \mathsf{X} \arrow[r, shift right, "R"'] \arrow[r, phantom, "\perp"] & \mathsf{A} \arrow[l, shift right, "L"'] \end{tikzcd}$$

such that L preserves finite limits. This definition generalizes the notion that a continuous map $f \colon X \to A$ between spaces induces the direct image $f_* \colon \operatorname{Sh}(X) \to \operatorname{Sh}(A)$ and inverse image $f^* \colon \operatorname{Sh}(A) \to \operatorname{Sh}(X)$ functors between categories of sheaves.

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 any cospan in X of the form $La \to x \leftarrow Lb$.

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 $La \leftarrow x +_{Lb} y \rightarrow Lc$.

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

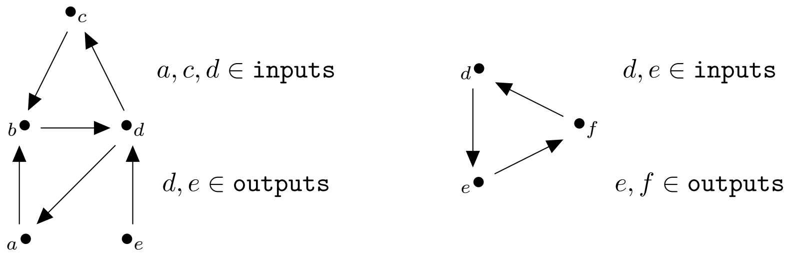
Definition 3.2. Fix a geometric morphism $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. Denote by ${}_L\mathsf{Csp}$ the category whose objects are those of A and whose arrows from a to b are (isomorphism classes of) structured cospans $La \to x \leftarrow Lb$.

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

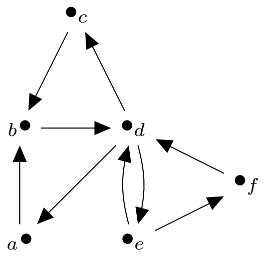
Example 3.3. Consider the geometric morphism

$$\begin{tikzcd} \mathsf{Graph} \arrow[r, bend right, "R"'] \arrow[r, phantom, "\perp"] & \mathsf{Set} \arrow[l, bend right, "L"'] \end{tikzcd}$$

where L is the discrete graph functor and R forgets the graph edges. Structured cospans for this adjunction are open graphs. A graph becomes open when equipped with two subset 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 open graph



$a, c, d \in$ `inputs`

$e, f \in$ `outputs`

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 ${}_L\mathsf{Csp}$. Thus our immediate task 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

$$\begin{tikzcd}
La \arrow[r, dash] \arrow[d, dash, "Lf"'] & x \arrow[r, dash] \arrow[d, dash, "g"'] & Lb \arrow[d, "Lh"] \\
Lc \arrow[r, dash] & y \arrow[r, dash] & Ld
\end{tikzcd}$$

Theorem 3.5. *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

$$\begin{tikzcd}
w \arrow[r] \arrow[d, "f"'] & Ra \arrow[d, "Rg"'] & x \arrow[l] \arrow[d, "h"'] \\
y \arrow[r] & Rb & z \arrow[l]
\end{tikzcd}$$

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. Hence, ΔR is a right adjoint making $(\mathsf{A} \times \mathsf{A} \downarrow \Delta R)$ 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

$$\begin{tikzcd}
\mathsf{X} \arrow[r, shift right, "R"'] \arrow[r, phantom, "\perp"] \arrow[d, shift left, "G"] & \mathsf{A} \arrow[l, shift right, "L"'] \arrow[d, leftrightarrow, shift right, "G'"'] \\
\mathsf{X}' \arrow[r, shift left, "R'"] \arrow[r, phantom, "\top"] \arrow[u, shift left, "F"] \arrow[u, phantom, shift left, "\dashv"] & \mathsf{A}' \arrow[l, shift left, "L'"] \arrow[u, shift right, "F'"'] \arrow[u, phantom, shift right, "\vdash"]
\end{tikzcd}
\xmapsto{{}_{(-)}\mathsf{StrCsp}}
\begin{tikzcd}
{}_L\mathsf{StrCsp} \arrow[r, shift left, "\Theta"] \arrow[r, phantom, "\perp"] & {}_{L'}\mathsf{StrCsp} \arrow[l, shift left, "\Theta'"]
\end{tikzcd}$$

which is in turn given by

$$\begin{tikzcd}
La \arrow[r, dash, "m"] \arrow[d, dash, "Lf"'] & x \arrow[r, dash, "n"] \arrow[d, dash, "g"'] & Lb \arrow[d, "Lh"'] \\
Lc \arrow[r, dash, "o"] & y \arrow[r, dash, "p"] & Ld
\end{tikzcd}
\overset{\Theta}{\longmapsto}
\begin{tikzcd}
L'G'a \arrow[r, "Gm"] \arrow[d, "L'G'f"'] & Gx \arrow[d, "Gg"'] & L'G'b \arrow[l, "Gn"'] \arrow[d, "L'G'h"'] \\
L'G'c \arrow[r, "Go"] & Gy & L'G'd \arrow[l, "Gp"']
\end{tikzcd}$$

and

$$\begin{tikzcd}
L'a' \arrow[r, "m'"] \arrow[d, "L'f'"'] & x' \arrow[d, "g'"'] & L'b' \arrow[l, "n'"'] \arrow[d, "L'h'"'] \\
L'c' \arrow[r, "o'"] & y' & L'd' \arrow[l, "p'"']
\end{tikzcd}
\overset{\Theta'}{\longmapsto}
\begin{tikzcd}
LF'a' \arrow[r, "Fm'"] \arrow[d, "LF'f'"'] & Fx' \arrow[d, "Fg'"'] & LF'b' \arrow[l, "Fn'"'] \arrow[d, "LF'h'"'] \\
LF'c' \arrow[r, "Fo'"] & Fy' & LF'd' \arrow[l, "Fp'"']
\end{tikzcd}$$

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

$$\begin{tikzcd}
\mathsf{X} \arrow[r, "F"] \arrow[d, bend left, "R"] & \mathsf{X}' \arrow[d, bend left, "R'"] \\
\mathsf{A} \arrow[u, bend left, "L"] \arrow[u, phantom, "\dashv"] \arrow[r, swap, "G"] & \mathsf{A}' \arrow[u, bend left, "L'"] \arrow[u, phantom, "\dashv"]
\end{tikzcd}$$

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 of structured cospans and their morphisms ${}_L\mathsf{StrCsp}$ is a topos. Lack and Sobocinski's aforementioned work ensures that structured cospans have a nice theory of rewriting [8, 9]. In this section, we develop this theory.

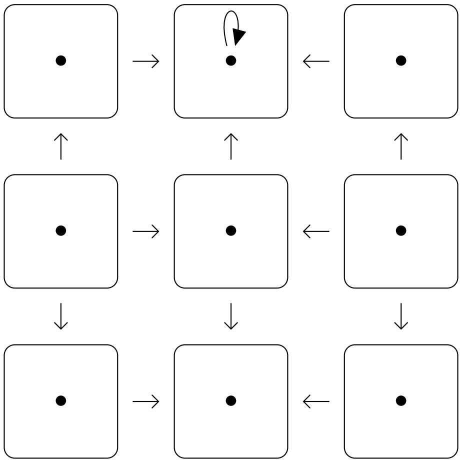
Definition 4.1. A **rewrite rule of structured cospans** is an isomorphism class of spans of structured cospans of the form

$$\begin{tikzcd}
La \arrow[r] & x & Lb \arrow[l] \\
Lc \arrow[u, "\cong"] \arrow[d, "\cong"'] \arrow[r] & y \arrow[u, tail] & Ld \arrow[u, "\cong"'] \arrow[d, "\cong"] \arrow[l] \\
Le \arrow[r] & z \arrow[u, tail] & Lf \arrow[l]
\end{tikzcd}$$

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 bottom. 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 more substantial difference is that we require that structured cospan rewriting does not affect the inputs and outputs. We accomplish this by including the isomorphisms on the outside, vertical arrows. This constraint ensures that rewriting interacts nicely with structured cospan composition.

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 ignore loops in our model, we can introduce the rule



(3)

which removes a loop on a single node that is both an input and output. Before we can apply this rule, we have some theory to develop.

For a fixed geometric morphism, there is a double category that contains all possible rewrites as squares.

Definition 4.2. 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 rewrites.

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. However, this has been shown in previous work [4, Lem. 4.2, Lem. 4.3].

4.1. **Structured cospan grammars.** In Definition 2.1, we defined a category Gram. An object in Gram is a topos paired with a set of rewrite rules and an arrow in Gram is a rule-preserving functor. Our interest now shifts in the subcategory of Gram that contains only 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) such that if the rewrite rule

$$\begin{tikzcd}
La \arrow[r] & x & Lb \arrow[l] \\
La' \arrow[r] \arrow[u] \arrow[d] & y \arrow[u] \arrow[d] & Lb' \arrow[l] \arrow[u] \arrow[d] \\
La'' \arrow[r] & z & Lb'' \arrow[l]
\end{tikzcd}$$

is in P, then its image under F is in P'. Note that this image, because of how we defined a morphism of structured cospan categories, is equal to the rule

$$\begin{tikzcd}
L'Ga \arrow[r] & Fx & L'Gb \arrow[l] \\
L'Ga' \arrow[r] \arrow[u] \arrow[d] & Fy \arrow[u] \arrow[d] & L'Gb' \arrow[l] \arrow[u] \arrow[d] \\
L'Ga'' \arrow[r] & Fz & L'Gb'' \arrow[l]
\end{tikzcd}$$

.

To analyze a structured cospan grammar, we want to look at its rewrite relation which, as in Section 2, is the reflexive and transitive closure of the relation $\rightsquigarrow$ induced by derived rules. And although the definition of the rewrite relation has not changed, we require a brief detour to ensure that derived rules in our new context are, in fact, rewrite rules of structured cospans.

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

$$\begin{tikzcd}
 & & Lb \arrow[rrr] \arrow[dl] \arrow[ddd] & & & Lb'' \arrow[ddd] \\
 & x \arrow[rrr] \arrow[ddd] & & & x'' \arrow[ur] \arrow[ddd] & \\
La \arrow[ur] \arrow[rrr] \arrow[ddd] & & & La'' \arrow[ur] \arrow[ddd] & & \\
 & & Lb' \arrow[rrr] & & & Lb' +_{Lb} Lb'' \arrow[dl] \\
 & x' \arrow[ur] \arrow[rrr] & & & x' +_x x'' & \\
La' \arrow[ur] \arrow[rrr] & & & La' +_{La} La'' \arrow[ur] & &
\end{tikzcd} \tag{4}$$

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 (5) 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

$$\begin{tikzcd}
La' \arrow[r] & x' & Lb' \arrow[l] \\
La \arrow[u, "\cong"] \arrow[d, "\cong"'] \arrow[r] & x \arrow[u, tail] \arrow[d, tail] & Lb \arrow[l] \arrow[u, "\cong"'] \arrow[d, "\cong"] \\
La'' \arrow[r] & x'' & Lb'' \arrow[l]
\end{tikzcd}$$

and morphism

$$\begin{tikzcd}
La' \arrow[r] \arrow[d] & x' \arrow[d] & Lb' \arrow[l] \arrow[d] \\
Lc' \arrow[r] & y' & Ld' \arrow[l]
\end{tikzcd}$$

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

$$\begin{tikzcd}
& & Lb' \arrow[dl] \arrow[ddd] & & & Lb \arrow[lll] \arrow[rrr] \arrow[dl] \arrow[ddd] & & & Lb'' \arrow[dl] \arrow[ddd] \\
& x' \arrow[ddd] & & & x \arrow[lll] \arrow[rrr] \arrow[ddd] & & & x'' \arrow[ddd] & \\
La' \arrow[ur] \arrow[ddd] & & & La \arrow[lll] \arrow[rrr] \arrow[ur] \arrow[ddd] & & & La'' \arrow[ur] \arrow[ddd] & & \\
& & Ld' \arrow[dl] & & & Ld \arrow[lll] \arrow[rrr] \arrow[dl] & & & Ld'' \arrow[dl] \\
& y' & & & y \arrow[lll] \arrow[rrr] & & & y'' & \\
Lc' \arrow[ur] & & & Lc \arrow[lll] \arrow[rrr] \arrow[ur] & & & Lc'' \arrow[ur] & &
\end{tikzcd} \tag{5}$$

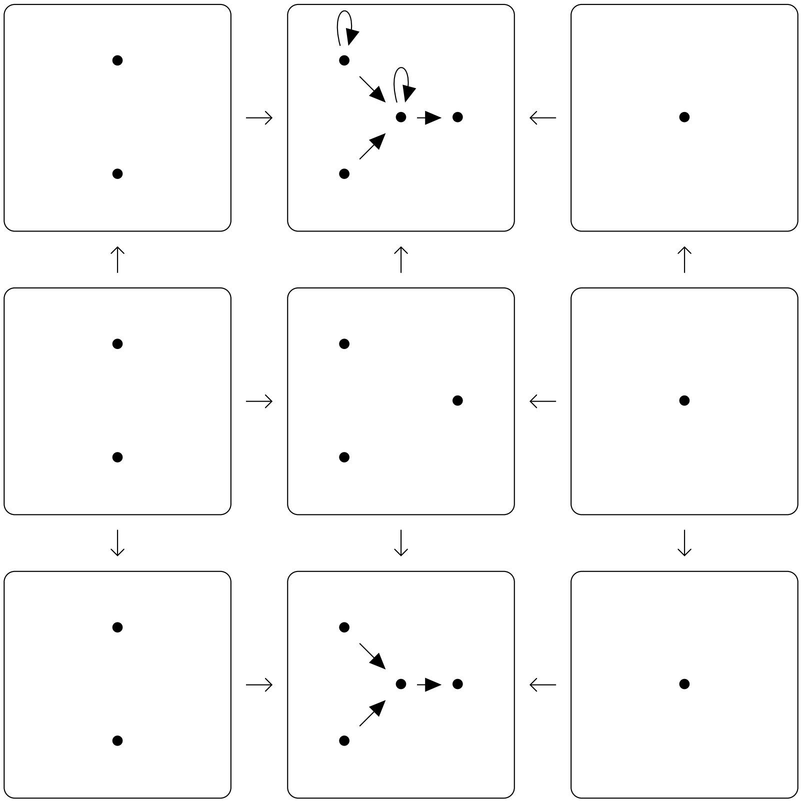
The bottom face is a rule because pushouts in a topos preserve monos and epis which, because topoi are balanced, implies that pushouts also preserve isomorphisms. In short, derived rules are rules.

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 might try to encode the rewrite relation as arrows in a category like we did with the lanugage functor Lang: $\mathsf{Gram} \to \mathsf{Cat}$ in Section 2. However, we have an added layer of composition to consider given that we are working with structured cospans. Therefore, we give our structured cospan language functor the type Lang: $\mathsf{StrCspGram} \to \mathsf{DblCat}$ where the codomain is the category of double categories and their functors. The double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ accommodates the structured cospans as horizontal arrows and structured cospan rewrites as squares. To help visualize Lang before giving its definition, we sketch a simple example.

Example 4.3. Starting with the discrete graph geometric morphism,

$$\begin{tikzcd}
\mathsf{RGraph} \arrow[r, phantom, "\perp"] \arrow[r, shift right, "R"'] & \mathsf{Set} \arrow[l, shift right, "L"']
\end{tikzcd}$$

for which ${}_L\mathsf{StrCsp}$ is the category of open graphs, consider a grammar P comprising only Rule 3. Then $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ 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, $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$ contains the square



The following theorem constructs the language functor that encodes the rewrites relation as squares in a double category.

Theorem 4.4. *Let $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ and $L' \dashv R' \colon \mathsf{X}' \to \mathsf{A}'$ be geometric morphisms.*

Given a structured cospan grammar $({}_L\mathsf{StrCsp}, P)$*, 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 structured cospans from* ${}_L\mathsf{StrCsp}$*, and squares are generated by the relation* $x \rightsquigarrow y$*.*

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. To each structured cospan grammar $({}_L\mathsf{StrCsp}, P)$, associate the set dP of rules derived from P. 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, this language functor provides a sound encoding of the rewrite relation into the squares of a double category.

Theorem 4.5. *Let* $({}_L\mathsf{StrCsp}, P)$ *be a structured cospan grammar. If* $(La \leftarrow x \to Lb) \rightsquigarrow^* (La' \leftarrow x' \to Lb')$*, then there is a square*

$$\begin{tikzcd}
La \arrow[r] & x & Lb \arrow[l] \\
Lc \arrow[r] \arrow[u] \arrow[d] & y \arrow[u] \arrow[d] & Ld \arrow[l] \arrow[u] \arrow[d] \\
La' \arrow[r] & x' & Lb' \arrow[l]
\end{tikzcd}$$

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

We do not expect Lang to provide a complete encoding because the compositionality of structured cospans provides a richer structure. We argue that, actually, the rewrite relation is not the morally correct semantics to study in the case of structured cospans. Indeed, Gadducci and Heckel state the inductive perspective of the rewrite relation is achieved by taking a closure of $\rightsquigarrow$. However, this closure does not consider the possible compositional nature of the objects being rewritten. Whereas, the Lang construction does close the rewriting along rewriting and object compositionality.

5. Expressiveness of grammars

In general, for any rewrite rule $\ell \leftarrow k \rightarrow r$ in general, there is no constraint on the value of k. Despite our interpretation that it is a subobject of ℓ and r that remains fixed through the rule's application, it can actually be any object fitting into the span. This freedom increased the difficulty of analysing grammars, so we devise a more restrained alternative. Specifically, we ask that k be *discrete*. In this section, we make the term 'discrete' precise and show that a grammar and its discretized variant induce the same rewrite relation. This result generalizes a characterization of discrete graph grammars given by Ehrig, et. al. [5, Prop. 3.3].

Experts in topos theory may recognize 'discreteness' from the flat modality on a local topos. However, we avoid the lengthy detour required to discuss what a 'flat modality' and a 'local topos' actually are because it would not add to our story (curious readers can find this information elsewhere [7, Ch. C3.6]). To avoid this detour, we define a 'discrete comonad'.

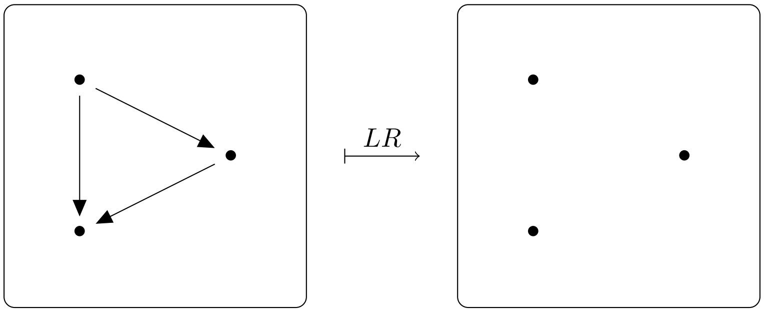
Definition 5.1 (Discrete comonad)**.** A comonad on a topos is called **discrete** if its counit is monic. We use $\flat$ to denote a discrete comonad.

Such a comonad acts on an object by returning a particular subobject which, in our systems interpretation, we take to be the interface of the system.

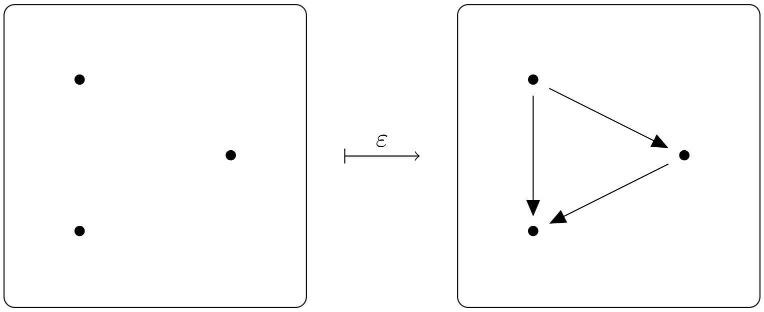
Example 5.2. The adjunction

$$\begin{tikzcd} \mathsf{RGraph} \arrow[r, bend right, "R"'] \arrow[r, phantom, "\perp"] & \mathsf{Set} \arrow[l, bend right, "L"'] \end{tikzcd}$$

induces the comonad LR on RGraph. Applying LR to a graph x returns the edgeless graph underlying x, for instance



The counit $\varepsilon_x \colon LRx \to x$ of the comonad LR includes the underlying edgeless graph LRx into the original graph x. For example,



From this example, the core property being generalized is best described using the systems interpretation. Given a geometric morphism $L \dashv R\colon \mathsf{X} \to \mathsf{A}$ that induces a discrete comonad LR, a system x is sent to a sub-system LRx comprising the largest subobject of x that can serve as its interface. The counit simply includes this interface into the system.

Why introduce discrete comonads into this story? We use them to control the form of our grammars. In particular, we can *discretize* any rewrite rule $\ell \leftarrow k \rightarrow r$ by replacing k with $\flat k$.

Definition 5.3 (Discrete grammar)**.** Given a grammar (T, P), define $P_\flat$ as the set containing

$$\ell \leftarrow k \leftarrow \flat k \rightarrow k \rightarrow r$$

for each rule $\ell \leftarrow k \rightarrow r$ 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 rewriting relation when, for each rule $\ell \leftarrow k \rightarrow r$ 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. Indeed, 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 either the over-category $T \downarrow k$ has 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 combine all ingredients from this section—the discrete comonad, the discrete grammar, and subobject lattice—into a result on the expressiveness on discrete grammars. 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 rewriting relation for (T, P) equals the rewriting relation for the underlying discrete grammar $(\mathsf{T}, P_\flat)$.*

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

$$\begin{tikzcd}
\ell \arrow[d] & k \arrow[l] \arrow[r] \arrow[d] & r \arrow[d] \\
g \arrow[ur, phantom, "\urcorner", very near start] & d \arrow[l] \arrow[r] & h \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{6}$$

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

$$\begin{tikzcd}
k \arrow[d] & \flat k \arrow[l, "\varepsilon"'] \\
d &
\end{tikzcd}$$

Define

$$w \coloneqq \bigwedge \{z \colon z \vee k = d\} \vee \flat k,$$

This comes with inclusions $\flat k \to w$ and $w \to d$. This w exists because $\mathrm{Sub}(k)$ has all meets. Note that $w \vee k = d$ and $w \wedge k = \flat k$ which means that

$$\begin{tikzcd}
k \arrow[d] & \flat k \arrow[l] \arrow[d] \\
d \arrow[ur, phantom, "\urcorner", very near start] & w \arrow[l]
\end{tikzcd}$$

is a pushout. It follows that there is a derivation

$$\begin{tikzcd}
\ell \arrow[d] & k \arrow[l] \arrow[d] & \flat k \arrow[l] \arrow[r] \arrow[d] & k \arrow[r] \arrow[d] & r \arrow[d] \\
g \arrow[ur, phantom, "\urcorner", very near start] & d \arrow[l] \arrow[ur, phantom, "\urcorner", very near start] & w \arrow[l] \arrow[r] & d \arrow[r] \arrow[ul, phantom, "\ulcorner", very near start] & h \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{7}$$

with respect to $P_\flat$ because, the top row is a rule in $P_\flat$. Therefore, $g \rightsquigarrow h$ via P in Diagram (6) implies that $g \rightsquigarrow^* h$ via $P_\flat$ as shown in Diagram (7).

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

$$\begin{tikzcd}
\ell \arrow[d, "m"'] & \flat k \arrow[l] \arrow[r] \arrow[d, "\theta"'] & r \arrow[d, "m'"] \\
g \arrow[ur, phantom, very near start, "\urcorner"] & d \arrow[l, "\psi"] \arrow[r] & h \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd} \tag{8}$$

By construction of $P_\flat$, the rule $\ell \leftarrow \flat k \rightarrow r$ in $P_\flat$ was induced from a rule

$$\ell \xleftarrow{\tau} k \rightarrow r$$

in P, meaning that the map $\flat k \rightarrow \ell$ factors through τ. Next, define d' to be the pushout of the diagram

$$\begin{tikzcd}
\flat k \arrow[r, "\varepsilon"] \arrow[d, "\theta"'] & k \arrow[d, "\widehat{\theta}"] \\
d \arrow[r, "\widehat{\varepsilon}"'] & d' \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

By invoking the universal property of this pushout with the maps

$$\psi \colon d \rightarrow g \quad \text{and} \quad m\tau \colon k \rightarrow \ell \rightarrow g,$$

we get a canonical map $d' \rightarrow g$ that we can fit into a commuting diagram

$$\begin{tikzcd}
& \flat k \arrow[dl] \arrow[dr, "\varepsilon"] \arrow[dd, "\theta"' near end] & \\
\ell \arrow[dd, "m"'] & & k \arrow[ll, "\tau"' near start, crossing over] \arrow[dd, "\widehat{\theta}"] \\
& d \arrow[dl, "\psi"] \arrow[dr, "\widehat{\varepsilon}"'] & \\
g \arrow[ur, phantom, very near start, "\urcorner"] & & d' \arrow[ll] \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

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

$$\begin{tikzcd}
k \arrow[r] \arrow[d] & r \arrow[d] \\
d' \arrow[r] & h \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

is a pushout. Sticking these two pushouts together

$$\begin{tikzcd}
\ell \arrow[d, "m"'] & k \arrow[l] \arrow[r] \arrow[d, "f"'] & r \arrow[d, "m'"] \\
g \arrow[ur, phantom, "\urcorner", very near start] & d' \arrow[l] \arrow[r] & h \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

shows that $g \rightsquigarrow h$ arises from P.

Because the relation $\rightsquigarrow$ is the same for P and $P_\flat$, it follows that $\rightsquigarrow^*$ is also the same as claimed. □

6. An inductive view of rewriting in a topos

In this section, we use structured cospans to introduce an inductive method of building the rewriting relation for a grammar (X, P). To apply this method, several things must be true of X. We discuss those specifics below. This method is based on work by Gadducci and Heckel [6] who created an inductive method of generating the rewriting relation for a graph grammar, that is when $\mathsf{T} \coloneqq \mathsf{Graph}$. Our work closely follows the path they set. In the systems interpretation, this section is about generating the rewriting relation of closed systems using open systems.

We start by formalizing closed systems 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 an arrow as a composite of arrows

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

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 rewriting 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

$$\begin{tikzcd} \mathsf{X} \arrow[r, shift right, "R"'] \arrow[r, phantom, "\perp"] & \mathsf{A} \arrow[l, shift right, "L"'] \end{tikzcd}$$

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

$$\begin{tikzcd} L0 \arrow[r] & \ell & LRk \arrow[l] \\ L0 \arrow[u] \arrow[d] \arrow[r] & LRk \arrow[u] \arrow[d] & LRk \arrow[l] \arrow[u] \arrow[d] \\ L0 \arrow[r] & r & LRk \arrow[l] \end{tikzcd} \tag{9}$$

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

$$\begin{tikzcd}
\ell_1 \arrow[d] & k_1 \arrow[l] \arrow[r] \arrow[d] & r_1 \arrow[dr] & & \ell_2 \arrow[dl] & k_2 \arrow[l] \arrow[r] \arrow[d] & r_2 \arrow[d] \arrow[dr, phantom, "\cdots"] & \ell_n \arrow[d] & k_n \arrow[l] \arrow[r] \arrow[d] & r_n \arrow[d] \\
x \arrow[ur, phantom, "\urcorner", very near start] & d_1 \arrow[l] \arrow[rr] & & w_1 \arrow[ul, phantom, "\ulcorner", very near start] \arrow[ur, phantom, "\urcorner", very near start] & & d_2 \arrow[ll] \arrow[r] & w_2 \arrow[ul, phantom, "\ulcorner", very near start] & w_{n-1} \arrow[ur, phantom, "\urcorner", very near start] & d_n \arrow[l] \arrow[r] & y \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

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

$$\begin{tikzcd}
\ell'_1 \arrow[d] & k'_1 \arrow[l] \arrow[r] \arrow[d] & r'_1 \arrow[dr] & & \ell'_2 \arrow[dl] & k'_2 \arrow[l] \arrow[r] \arrow[d] & r'_2 \arrow[d] \arrow[dr, phantom, "\cdots"] & \ell'_m \arrow[d] & k'_m \arrow[l] \arrow[r] \arrow[d] & r'_m \arrow[d] \\
x' \arrow[ur, phantom, "\urcorner", very near start] & d'_1 \arrow[l] \arrow[rr] & & w'_1 \arrow[ul, phantom, "\ulcorner", very near start] \arrow[ur, phantom, "\urcorner", very near start] & & d'_2 \arrow[ll] \arrow[r] & w'_2 \arrow[ul, phantom, "\ulcorner", very near start] & w'_{m-1} \arrow[ur, phantom, "\urcorner", very near start] & d'_m \arrow[l] \arrow[r] & y' \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

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

$$\begin{tikzcd}
\ell_1 \arrow[d] & k_1 \arrow[l] \arrow[r] \arrow[d] & r1 \arrow[d] \arrow[dr, phantom, "\cdots"] & r_n \arrow[dr] & & \ell'_1 \arrow[dl] & k'_1 \arrow[l] \arrow[r] \arrow[d] & r'_1 \arrow[d] \arrow[dr, phantom, "\cdots"] & k'_m \arrow[r] \arrow[d] & r'_m \arrow[d] \\
x + x' & d_1 + x' \arrow[l] \arrow[r] & w_1 + x' & & y + x' & & y + d'_1 \arrow[ll] \arrow[r] & y + w'_1 & y + d'_m \arrow[r] & y + y'
\end{tikzcd}$$

□

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 rewriting relation for a grammar* (X, P) *if and only if there is a square*

$$\begin{tikzcd}
LR0 \arrow[r] & g & LR0 \arrow[l] \\
LR0 \arrow[r] \arrow[u] \arrow[d] & d \arrow[u] \arrow[d] & LR0 \arrow[l] \arrow[u] \arrow[d] \\
LR0 \arrow[r] & h & LR0 \arrow[l]
\end{tikzcd}$$

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

$$\begin{tikzcd}
\ell \arrow[d] & LRk \arrow[l] \arrow[r] \arrow[d] \arrow[dl, phantom, very near end, "\urcorner"] \arrow[dr, phantom, very near end, "\ulcorner"] & r \arrow[d] \\
g & d \arrow[l] \arrow[r] & h
\end{tikzcd}$$

then the desired square is the horizontal composition of

$$\begin{tikzcd}
L0 \arrow[r] & \ell & LRk \arrow[l] \arrow[r] & d & L0 \arrow[l] \\
L0 \arrow[r] \arrow[u] \arrow[d] & LRk \arrow[u] \arrow[d] & LRk \arrow[l] \arrow[r] \arrow[u] \arrow[d] & d \arrow[u] \arrow[d] & L0 \arrow[l] \arrow[u] \arrow[d] \\
L0 \arrow[r] & r & LRk \arrow[l] \arrow[r] & d & L0 \arrow[l]
\end{tikzcd}$$

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

$$\begin{tikzcd}
L0 \arrow[r] & g & L0 \arrow[l] \\
L0 \arrow[r] \arrow[u] \arrow[d] & d \arrow[u] \arrow[d] & L0 \arrow[l] \arrow[u] \arrow[d] \\
L0 \arrow[r] & h & L0 \arrow[l] \\
L0 \arrow[r] \arrow[u] \arrow[d] & e \arrow[u] \arrow[d] & L0 \arrow[l] \arrow[u] \arrow[d] \\
L0 \arrow[r] & j & L0 \arrow[l]
\end{tikzcd}$$

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

$$\begin{tikzcd}
L0 & w \arrow[l] \arrow[r] & L0 \\
L0 \arrow[u] \arrow[d] & x \arrow[l] \arrow[r] \arrow[u] \arrow[d] & L0 \arrow[u] \arrow[d] \\
L0 & y \arrow[l] \arrow[r] & L0
\end{tikzcd}$$

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

$$\begin{tikzcd}
L0 & y \arrow[l] \arrow[r] & L0 \\
L0 \arrow[u] \arrow[d] & L0 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & L0 \arrow[u] \arrow[d] \\
L0 & z \arrow[l] \arrow[r] & L0
\end{tikzcd}$$

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

$$\begin{tikzcd}
L0 & w \arrow[l] \arrow[r] & L0 \\
L0 \arrow[u] \arrow[d] & L0 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & L0 \arrow[u] \arrow[d] \\
L0 & z \arrow[l] \arrow[r] & L0
\end{tikzcd}$$

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

$$\begin{tikzcd}
L0 & \ell \arrow[l] \arrow[r] & L0 \\
L0 \arrow[u] \arrow[d] & LRk \arrow[l] \arrow[r] \arrow[u] \arrow[d] & L0 \arrow[u] \arrow[d] \\
L0 & r \arrow[l] \arrow[r] & L0
\end{tikzcd}$$

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

$$\begin{tikzcd}
L0 & w+\ell \arrow[l] \arrow[r] & L0 \\
L0 \arrow[u] \arrow[d] & x+LRk \arrow[l, tail] \arrow[r, tail] \arrow[u] \arrow[d] & L0 \arrow[u] \arrow[d] \\
L0 & y+r \arrow[l] \arrow[r] & L0
\end{tikzcd}$$

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 $L0 \to x \leftarrow L0$ 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

$$\begin{tikzcd}
& x_1 & & x_2 & & & x_n & \\
L0 \arrow[ur] & & La_1 \arrow[ul] \arrow[ur] & & La_2 \arrow[ul] & \cdots & La_{n-1} \arrow[u] & L0 \arrow[ul]
\end{tikzcd}$$

Rewriting can be performed on each of these sub-systems

$$\begin{tikzcd}
L0 \arrow[r] & x_1 & La_1 \arrow[l] & & La_{n-1} \arrow[r] & x_n & L0 \arrow[l] \\
L0 \arrow[u, "\cong"] \arrow[d, "\cong"'] \arrow[r] & x'_1 \arrow[u, tail] \arrow[d, tail] & La'_1 \arrow[l] \arrow[u, "\cong"'] \arrow[d, "\cong"] & \cdots & La_{n-1} \arrow[r] \arrow[u, "\cong"] \arrow[d, "\cong"'] & x'_n \arrow[u, tail] \arrow[d, tail] & L0 \arrow[l] \arrow[u, "\cong"'] \arrow[d, "\cong"] \\
L0 \arrow[r] & x''_1 & La''_1 \arrow[l] & & La_{n-1} \arrow[r] & x''_n & L0 \arrow[l] \\
& \vdots & & & & \vdots & \\
L0 \arrow[r] & y_1 & La_1 \arrow[l] & & La_{n-1} \arrow[r] & y_n & L0 \arrow[l] \\
L0 \arrow[u, "\cong"] \arrow[d, "\cong"'] \arrow[r] & y'_1 \arrow[u, tail] \arrow[d, tail] & La_1 \arrow[l] \arrow[u, "\cong"'] \arrow[d, "\cong"] & \cdots & La_{n-1} \arrow[r] \arrow[u, "\cong"] \arrow[d, "\cong"'] & y'_n \arrow[u, tail] \arrow[d, tail] & L0 \arrow[l] \arrow[u, "\cong"'] \arrow[d, "\cong"] \\
L0 \arrow[r] & y''_1 & La_1 \arrow[l] & & La_{n-1} \arrow[r] & y''_n & L0 \arrow[l]
\end{tikzcd}$$

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

References

[1] John C. Baez and Kenny Courser. Structured cospans. *In Preperation.* (Referred to on page 2, 8.)

[2] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959. (Referred to on page 23.)

[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 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 14.)

[5] H. Ehrig, M. Pfender, and H. J. 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 7, 20, 23.)

[6] F. Gadducci and R. 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 26.)

[7] Peter T 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, 5, 12.)

[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 4, 12.)

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

Email address: dcicala@newhaven.edu

Department of Mathematics and Physics, University of New Haven