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

2. Rewriting in topoi

The topic of double pushout rewriting is well represented in the literature. Here, we cover the basics and set our notation and convention. Experts can skip to the next section. Novices have many suitable choices to supplement this review, though we suggest Lack and Sobocinski's axiomatization of rewriting in adhesive categories [6].

Date: October 17, 2019.

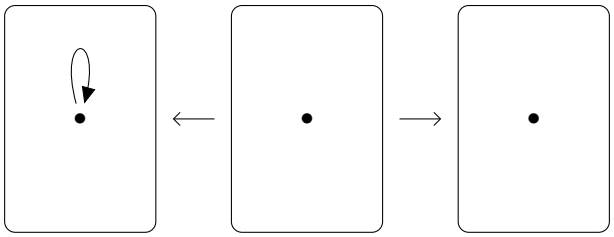
Rewriting starts with the notion of a **rewrite rule**, or simply **rule**. In its most general form, a rule is a span

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

in a topos. For us, the arrows are of the span are always monic and, whenever possible, unnamed.

The conceit of a rule is that r replaces ℓ while k identifies a subsystem of ℓ that remains fixed. For example, suppose we were modeling some system using graphs where self-loops are negligable. For example, consider a model of the internet as a graph with websites for nodes and links for edges. If we wished to ignore links from a websites to itself, we can introduce a rule that replaces a loop on a node with that node

(1)



To *apply* a rule $\ell \leftarrow k \rightarrow r$ to an object ℓ', we require an arrow $\ell \rightarrow \ell'$ such that there exists a **pushout complement**, an object d fitting into a pushout diagram

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

A pushout complement need not exist, but when it does and the map $k \rightarrow \ell$ is monic, then it is unique up to isomorphism [6, Lem. 15].

Each application of a rule begets a new rule.

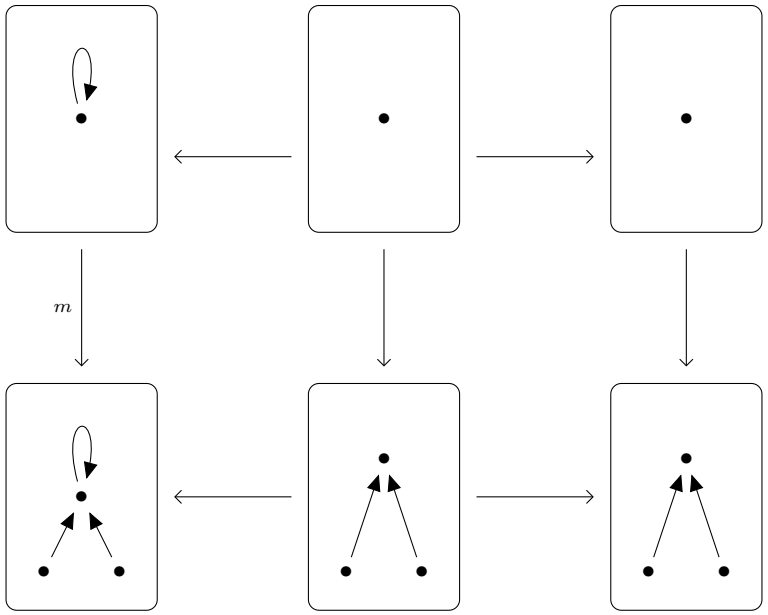
Definition 2.1 (Derived rule)**.** Applying a rule $\ell \leftarrow k \rightarrow r$ to an object g induces a **derived rule** $\ell' \leftarrow k' \rightarrow r'$ that fits into the bottom row of the double pushout diagram

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

The arrows of derived rule are also monic because pushouts in topoi preserve monics [**?**, Lem. 12]. Interpret this diagram as follows: the map $\ell \rightarrow \ell'$ selects a copy of ℓ inside ℓ' and that copy is replaced by r resulting in the new object r'.

To illustrate, let us return to the internet modeled using graphs and negligable self-loops. Then we can apply Rule (**??**) to any node with a loop. This application is captured with the double pushout diagram

(2)



The map m selects the self-loop and corresponding node in the bottom left graph. A pushout complement exists, meaning that the rule can be applied and so the loop

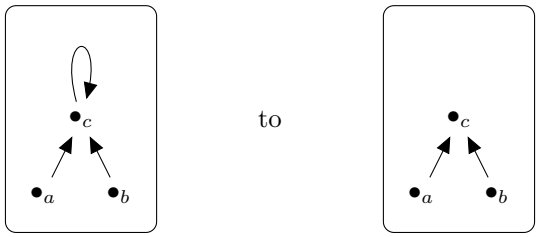
is removed. The result is the bottom right graph. The reader can check that the two squares are pushouts.

Specialists is a particular sort of system spend time understanding the its rewrite rules. Resistor circuits have parallel, series, and star rules. Other systems have their own versions. Therefore, we consider a system equipped with a set of rules.

Definition 2.2 (Grammar)**.** A topos T together with a finite set P of rules $\{\ell_j \leftarrow k_j \rightarrow r_j\}$ in T is a **grammar**. A morphisms 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 for each rule $\ell \xleftarrow{f} k \xrightarrow{g} r$ in P, the rule $F\ell \xleftarrow{Ff} Fk \xrightarrow{Fg} Fr$ is in Q. Together these form a category Gram.

Like a seed, from a grammar grows something entirely new called the *language*. The language contains all necessary information about the syntax of a system, so naturally, it is the thing to study. The potential complexity of a language makes this a difficult task, so we instead turn to a more accessible proxy called the *rewrite relation*. This relation captures the relationship between the many sytactical componenets of a language.

Constructing the rewrite relation requires several steps. A grammar (T, P) induces a relation $\rightsquigarrow$ on the objects of T defined by $g \rightsquigarrow h$ whenever there exists a rule $g \leftarrow d \rightarrow h$ derived from a production in P. For instance, the above double pushout diagram would relate



But $\rightsquigarrow$ is too small to capture the full behavior of the language. For one, it is not true in general that $g \rightsquigarrow g$ holds. Also, $\rightsquigarrow$ does not capture multi-step rewrites. That is, there may be derived rules witnessing $g \rightsquigarrow g'$ and $g' \rightsquigarrow g''$ but not a derived rule witnessing $g \rightsquigarrow g''$. We want to relate a pair of objects if one can be rewritten into another using a finite sequence of rules that are either in P or derived from a rule in P. Therefore, we define the **rewrite relation** to be the reflexive and transitive closure of $\rightsquigarrow$

After we introduce a grammar for structured cospans in Section **??**, we show that constructing it is functorial.

3. Structured cospans form a topos

Having discussed rewriting in a topos, in order to introduce the rewriting of structured cospans, we better ensure that they form a topos. Indeed, we do this after recalling their definition.

Baez and Courser introduced structured cospans [1] to serve as a syntax for open systems. We use the term 'system' broadly and avoid giving a precise definition. By 'open', we mean that compatible systems can be connected to form a larger system. For example, a vacuum cleaner can connected to the electrical grid via an electrical socket. Or, a pulley system can connect to a mechanical motor.

In order to form a topos out of structured cospans, we require stronger conditions than assumed by Baez and Courser.

Definition 3.1. Fix an adjunction

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

between (elementary) topoi with L preserving finite limits. An **L-structured cospan**, or simply **structured cospan**, is any cospan in X of the form $La \to x \leftarrow Lb$.

How does the structured cospan syntax map onto an open system? The apex x of the cospan represents the system itself. The legs of the cospan, $La \to x$ and $Lb \to x$, select the inputs La and outputs Lb of x. These legs endow structured cospans with the ability to connect together. Given $La \to x \leftarrow Lb$ and $Lb \to y \leftarrow Lc$, we form the *composite* structured cospan $La \to x +_{Lb} y \leftarrow Lc$. Interpret the pushout $x +_{Lb} y$ as an object obtained by glueing x to y along their common interface Lb. The reason we call $La \leftarrow x +_{Lb} y \to Lc$ the 'composite' is because a natural way to emphasize this connectability is by defining a category in which structured cospans are the arrows.

Definition 3.2. Given an adjunction

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

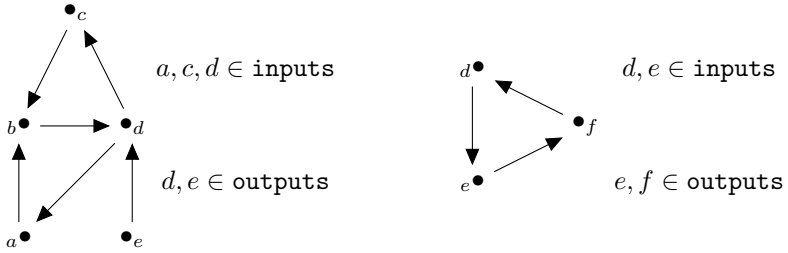
between (elementary) topoi with L preserving finite limits, denote by ${}_L\mathsf{Csp}$ the category whose objects are those of A and arrows from a to b are structured cospans $La \to x \leftarrow Lb$.

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

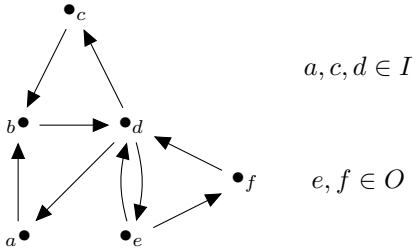
Example 3.3. Consider the adjunction

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

with L the discrete graph functor and R the functor that forgets the graph edges. Structured cospans for this adjunction are open graphs, that is graphs equipped with two subset of nodes, one set interpretted as inputs and the other as outputs. These "interface" sets need not be disjoint. For example, the pair of open graphs



are composed by glueing like-nodes together



Let us pause for a moment to recall that, in Section 2, we showed that rewriting operates on the *objects* of a topos, not the arrows. To comply with this, our immediate tasks become to define a category where structured cospans are objects and then show that category to be a topos.

Definition 3.4. Let $L \dashv R$ be an adjunction between topoi where L preserves finite limits. There is a category ${}_L\mathsf{StrCsp}$ whose objects are L-structured cospans and arrows from $La \to x \leftarrow Lb$ to $Lc \to y \leftarrow Ld$ are triple of arrows (f, g, h) fitting into the commuting diagram

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

Theorem 3.5. *Given an adjunction*

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

between topoi with L preserving finite limits, 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

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

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. Hence, ΔR is a right adjoint making $(\mathsf{A} \times \mathsf{A} \downarrow \Delta R)$ an Artin glueing [7], therefore a topos. □

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

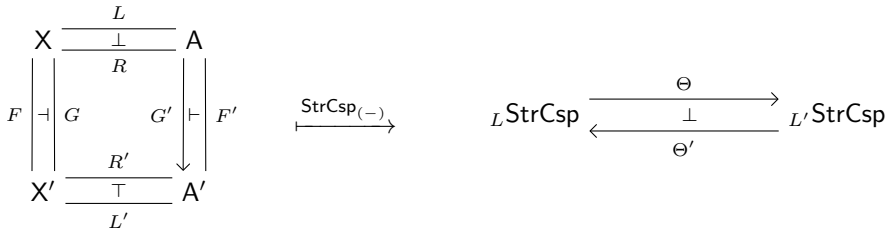
Theorem 3.6. *Denote by* Topos *the category whose objects are topoi and whose arrows from* X *to* A *are adjoint pairs*

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

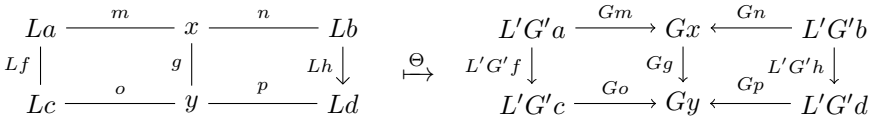
where L preserves finite limits. There is a functor

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

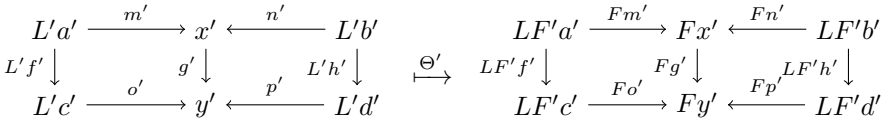
defined by



which is in turn given by



and



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

Denote the structured cospans

$$[m, n] \colon La \to x \leftarrow Lb$$

in ${}_{L}\mathsf{StrCsp}$ by ℓ and

$$[m', n'] \colon L'a' \to x' \leftarrow L'b'$$

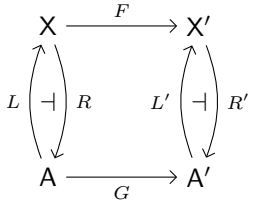
in ${}_{L'}\mathsf{StrCsp}$ by ℓ'. Denote the unit and counit for $F \dashv G$ by η, ε and for $F' \dashv G'$ by η', ε'. The assignments

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

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

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. □

With this theorem, we have highlighted categories of the form ${}_L\mathsf{StrCsp}$ where L runs through finite limit preserving left adjoints. These **structured cospan categories** fit into a 2-category **StrCsp**. A 1-arrow from ${}_L\mathsf{StrCsp} \to_{L'} \mathsf{StrCsp}$ is a 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



. A 2-arrow from $(F, G) \Rightarrow (F', G')$ is a pair of natural transformations $\alpha\colon F \Rightarrow F'$ and $\beta\colon G \to G'$.

4. Rewriting structured cospans

(Generalize gadducci/heckle's theorem. Place it into the ACT context.)

(require that L have monic counit from the start)

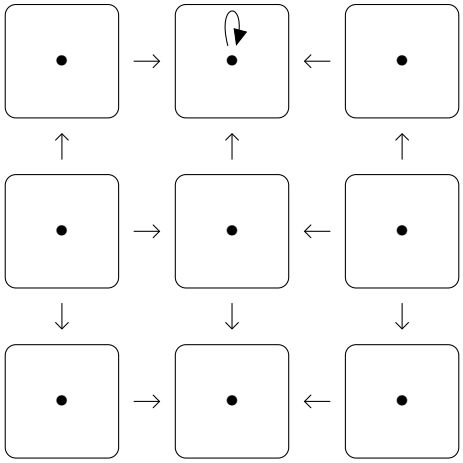
In Definition 2.2, we defined a category Gram where an object is a topos paired with a set of rewrite rules and an arrow $(\mathsf{T}, P) \to (\mathsf{T}', P')$ is a rule-preserving functor $\mathsf{T} \to \mathsf{T}'$. Our interest now lies in the subcategory of structured cospan grammars $\mathsf{StrCspGram}$ that is full on the grammars where T is a structured cospan category and P is a set of structured cospan rewrite rules.

It is on this category $\mathsf{StrCspGram}$ that we define the *language functor* $\mathrm{Lang}\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$ which encodes the rewrite relation to each grammar. The "language functor" is an appropriate name as it provides *(i)* the terms formed by connecting together open systems (instead of concatenating units of syntax in linguistics) and *(ii)* the rules governing how to interchange open systems (instead of parts of speech). To help visualize this, we sketch a simple example.

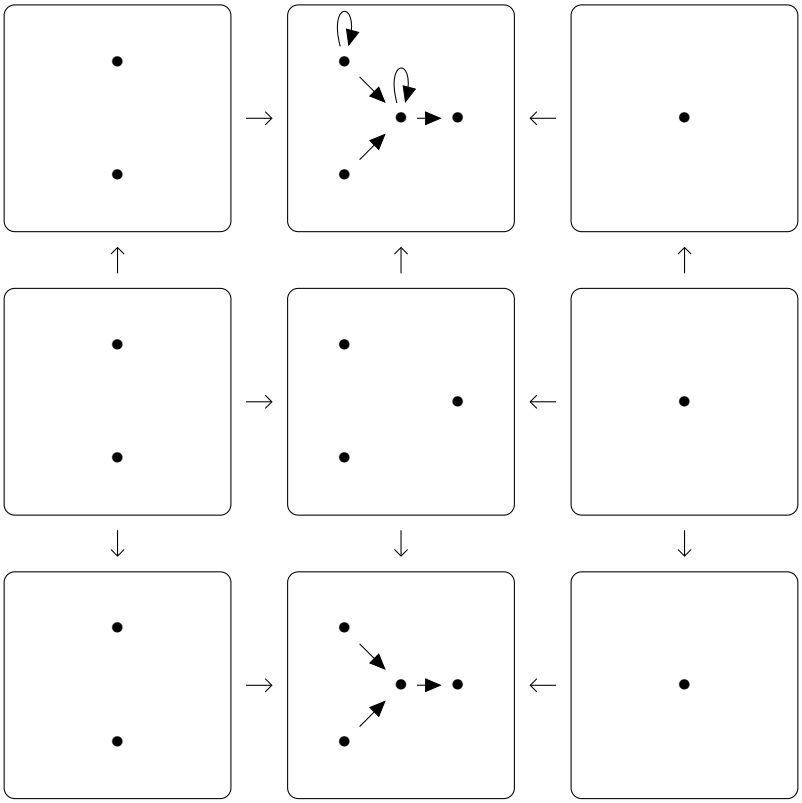
Example 4.1. Starting with the discrete graph adjunction

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

for which ${}_L\mathsf{StrCsp}$ is the category of open graphs, consider a grammar defined by the single rule



The language associated to this grammar consists of all open graphs. The rewrite relation $g \rightsquigarrow^* h$ means that we obtained h by removing loops from g. We illustrate this with the following square in the double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, P)$.



We construct Lang in two steps. First, we apply a functor $D\colon \mathsf{StrCspGram} \to \mathsf{StrCspGram}$ that sends a grammar $({}_L\mathsf{StrCsp}, P)$ to all of the rewrite rules derived from P. Second, we apply a functor $S\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$. that generates a double category on the squares obtained from the rewrite rules of a grammar $({}_L\mathsf{StrCsp}, P)$. In this way, we get the language of a grammar as a double category where the squares are the rewrite rules. The next lemma defines D and extracts some of its properties.

Lemma 4.2. *There is an idempotent functor $D\colon \mathsf{StrCspGram} \to \mathsf{StrCspGram}$ defined as follows. On objects define $D({}_L\mathsf{StrCsp}, P)$ to be the grammar $({}_L\mathsf{StrCsp}, P_D)$, where P_D consists of all rules $g \leftarrow h \to d$ witnessing the relation $g \rightsquigarrow h$ with respect*

to $({}_L\mathsf{StrCsp}, P)$. *On arrows, define* $DF\colon D({}_L\mathsf{StrCsp}, P) \to D({}_{L'}\mathsf{StrCsp}, Q)$ *to be* F. *Moreover, the identity on* $\mathsf{StrCspGram}$ *is a subfunctor of* D.

Proof. That $D({}_L\mathsf{StrCsp}, P)$ actually gives a grammar follows from the fact that pushouts respect monics in a topos [6, Lem. 12].

To show that D is idempotent, we show that for any grammar $({}_L\mathsf{StrCsp}, P)$, we have $D({}_L\mathsf{StrCsp}, P) = DD({}_L\mathsf{StrCsp}, P)$. Rules in $DD({}_L\mathsf{StrCsp}, P)$ appear in the bottom row of a double pushout diagram whose top row is a rule in $D({}_L\mathsf{StrCsp}, P)$, which in turn is the bottom row of a double pushout diagram whose top row is in $({}_L\mathsf{StrCsp}, P)$. Thus, a rule in $DD({}_L\mathsf{StrCsp}, P)$ is the bottom row of a double pushout diagram whose top row is in $({}_L\mathsf{StrCsp}, P)$. See Figure 1.

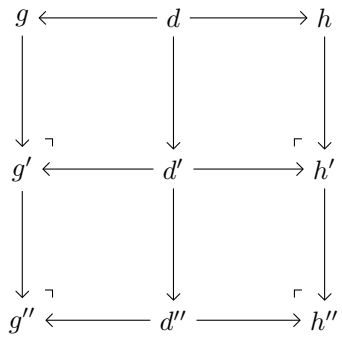


FIGURE 1. Stacked double pushout diagrams

The identity is a subfunctor of D because the identity on any rule $\ell \leftarrow k \to r$ in $({}_L\mathsf{StrCsp}, P)$ induces $\ell \rightsquigarrow r$. Hence there is a monomorphism

$$({}_L\mathsf{StrCsp}, P) \to D({}_L\mathsf{StrCsp}, P)$$

induced from the identity functor on ${}_L\mathsf{StrCsp}$. □

Plainly stated, D sends a grammar to a new grammar consisting of all derived rules. That D is idempotent means that a rule derived from P can be derived directly, hence multiple applications of D are unnecessary. That the identity is a subfunctor of D means that set of the derived rules P_D contains rules in P.

The next stage in defining Lang is to define $S\colon \mathsf{StrCspGram} \to \mathsf{DblCat}$. On objects, let $S({}_L\mathsf{StrCsp}, P)$ be the sub-double category of ${}_L\mathbb{S}\mathbf{trCsp}$ generated by the rules in P considered as squares. On arrows, S sends

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

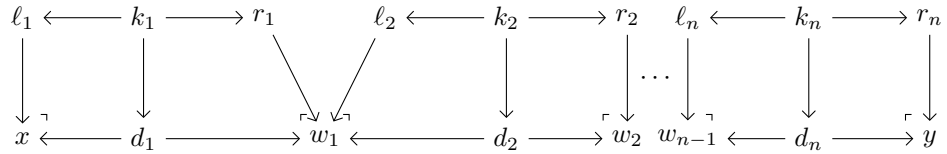
to the double functor defined that extends the mapping between the generators of $S({}_L\mathsf{StrCsp}, P)$ and $S({}_{L'}\mathsf{StrCsp}, P')$. This preserves composition because F preserves pullbacks and pushouts.

Definition 4.3. (Language of a grammar) The **language functor** is defined to be Lang $\coloneqq SD$.

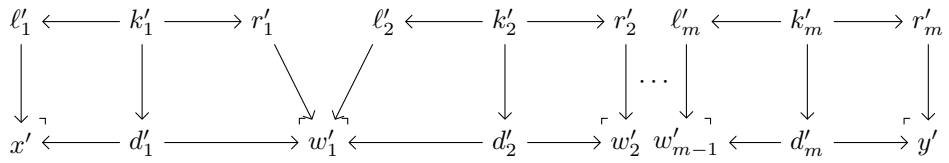
To witness the rewriting relation on a closed system as a square in a double category, we require this next lemma that formalizes the analogy between rewriting the disjoint union of systems and tensoring squares.

Lemma 4.4. *If $x \rightsquigarrow^* y$ and $x' \rightsquigarrow^* y'$, then $x + x' \rightsquigarrow^* y + y'$*

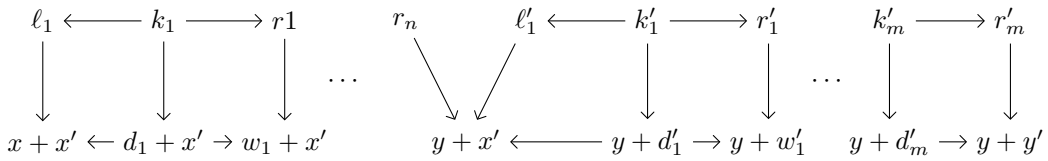
Proof. If the derivation $x \rightsquigarrow^* y$ comes from a string of double pushout diagrams



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



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



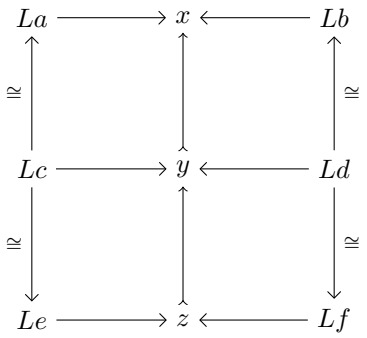
□

5. A double category of structured cospan rewrites

(Cover what a grammar is and organize them into a double category a la FineRewrite. Uncertain to include: interchange law, symmetric monoidal structure. Just point to thesis? Dno't call things FineRewrite, just Rewrite.)

Because ${}_L\mathsf{StrCsp}$, the category of structured cospans and their morphisms, is a topos, we are ensured that rewriting structured cospans have a nice theory of rewriting a lá adhesive categories [6]. The goal of this section is to define a double category whose squares are rewrites of structured cospans. The rough idea is that this double category, denoted ${}_L\mathbb{R}\mathbf{ewrite}$, has "interface" types for objects, structured cospans for horizontal arrows, isomorphisms of interface objects for vertical arrows, and rewrite rules of structured cospans for squares. We prove in Proposition **??** that ${}_L\mathbb{R}\mathbf{ewrite}$ actually is a double category. The first step to proving this is to ensure the rewrite rules are suitable squares for our double category, we define them as follows.

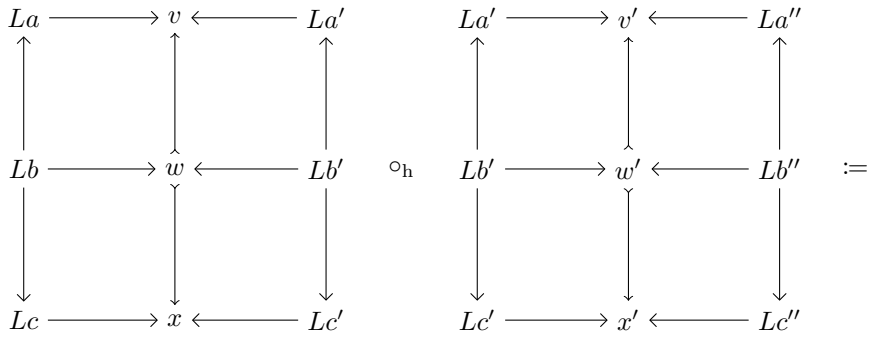
Definition 5.1 (Rewrite)**.** A **rewrite rule of structured cospans** is an isomorphism class of spans of structured cospans of the form

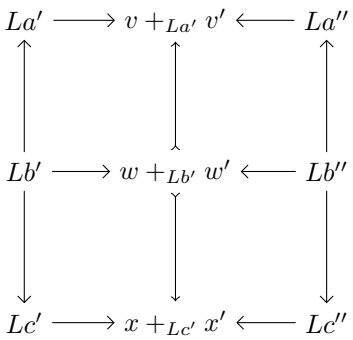


The marked arrows are monic.

Working with isomorphism classes is a technical point. The squares of a double category are subject to two composition operations. Below, we define horizontal composition using pushout, as is standard with cospan categories, and vertical composition uses pullback, as is standard in span categories. But pushouts and pullbacks are defined only up to isomorphism and there are no higher order arrows traversing the squares in a double category. Therefore, we must work with isomorphism classes of structured cospan rewrite rules.

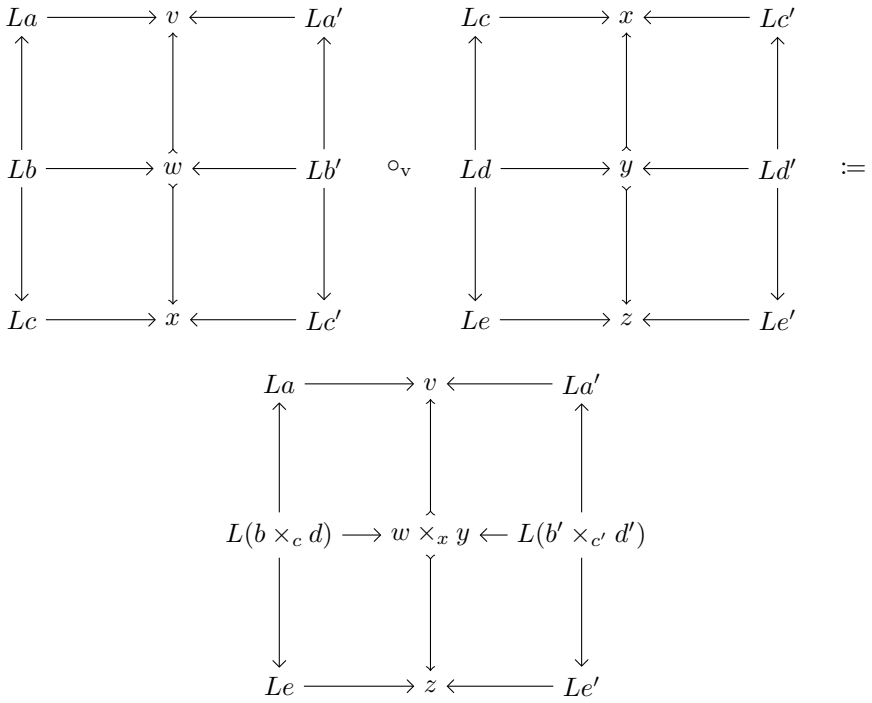
The **horizontal composition** of rewrite rules is given by





We interpret this as taking composable open systems, each undergoing a rewrite, and combining them into a single open system undergoing a single rewrite.

The **vertical composition** of rewrite rules is given by



We interpret this as an open system undergoing a two-step rewrite.

Here, we must make two technical points. The first is that horizontal and vertical composition are defined on representatives of isomorphism classes and one can show that operation is well-defined. The second is that we require the monic arrows to

be preserved, which we show in Proposition 5.3 below. To prove this proposition, we requires the following lemma.

Lemma 5.2. *The diagram*

$$\begin{array}{ccccc} x & \longleftarrow & y & \longrightarrow & z \\ \downarrow & & \downarrow \cong & & \downarrow \\ x' & \longleftarrow & y' & \longrightarrow & z' \end{array} \tag{3}$$

induces a pushout

$$\begin{array}{ccc} x + z & \xrightarrow{\rho} & x +_y z \\ {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle\gamma'} \\ x' + z' & \xrightarrow[\rho']{} & x' +_{y'} z \end{array} \tag{4}$$

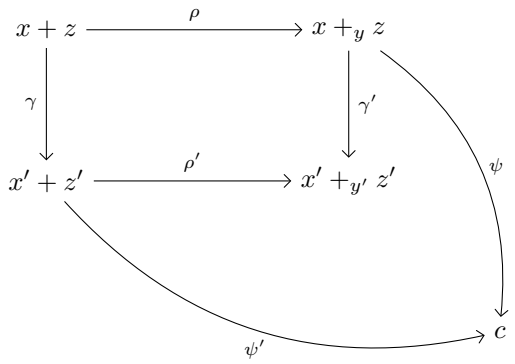
such that the canonical arrows γ and γ' are monic.

Proof. The universal property of coproducts implies that γ factors through $x' + z$ as in the diagram

$$\begin{array}{ccccc} x & \xrightarrow{\iota_x} & x + z & & \\ \downarrow & & \downarrow & & \\ x' & \xrightarrow[\iota_{x'}]{} & x' + z & \xleftarrow{\iota_z} & z \\ & & \downarrow & & \downarrow \\ & & x' + z' & \xleftarrow[\iota_{z'}]{} & z' \end{array}$$

It is straightforward to check that both squares are pushouts. By Lemma **??**, it follows that γ is monic.

Diagram 4 commutes because of the universal property of coproducts. To see that it is a pushout, arrange a cocone



(5)

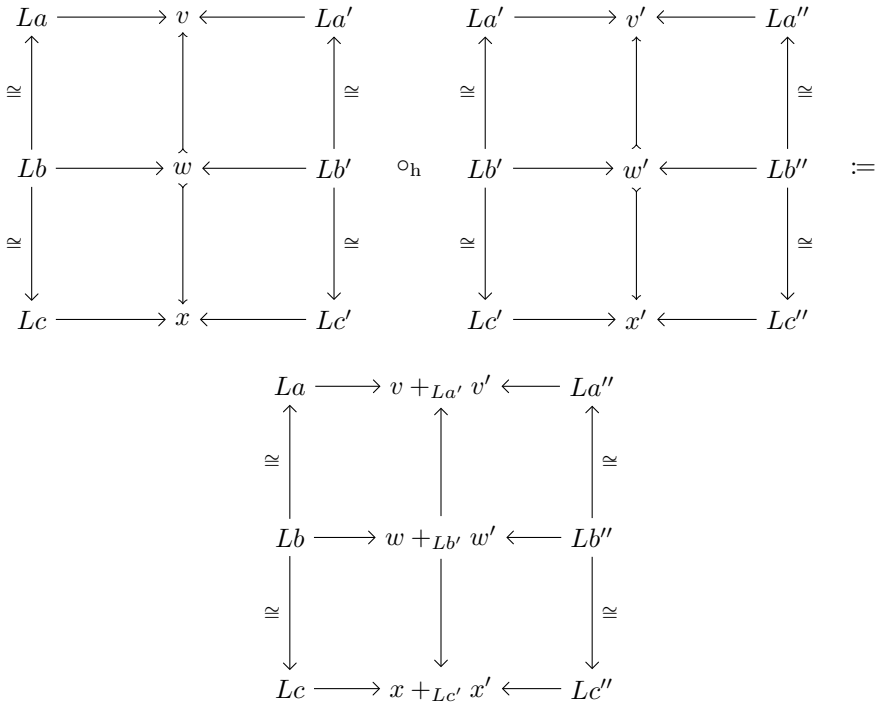
Denote by ι_x any map that includes x. Then $\psi'\iota_{x'}$, $\psi'\iota_{z'}$, and c form a cocone under the span $x' \leftarrow y' \rightarrow z'$ from the bottom face of Diagram 3. This induces the canonical map $\psi'' \colon x' +_{y'} z' \rightarrow c$. It follows that $\psi'\iota_{x'} = \psi''\rho'\iota_{x'}$ and $\psi'\iota_{z'} = \psi''\rho'\iota_{z'}$. Therefore $\psi' = \psi''\rho'$ by the universal property of coproducts.

Furthermore, $\psi\rho\iota_z$, $\psi\rho\iota_z$, and c form a cocone under the span $x \leftarrow y \rightarrow z$ on the top face of Diagram 3. then $\psi\rho\iota_x = \psi'\gamma\iota_x = \psi''\rho'\gamma\iota_x = \psi''\psi'\rho\iota_x$ and $\psi\rho\iota_z = \psi'\gamma\iota_z = \psi''\rho'\gamma\iota_z = \psi''\gamma'\rho\iota_z$ meaning that both ψ and $\psi''\psi'$ satisfy the canonical map $x +_y z \rightarrow d$. Hence $\psi = \psi''\psi'$.

The universality of ψ'' with respect to Diagram 5 follows from the universality of γ'' with respect to $x' +_{y'} z'$. □

Proposition 5.3. *Horizontal and vertical composition preserve rewrite rules of structured cospans.*

Proof. For the horizontal composition of rewrites



, the arrows $w +_{Le} x \to u +_{Lb} v$ and $w +_{Le} x \to y +_{Lh} z$ are monic. This is seen by applying Lemma 5.2 to the diagrams

The result for vertical composition holds because pullback preserves monomorphisms.

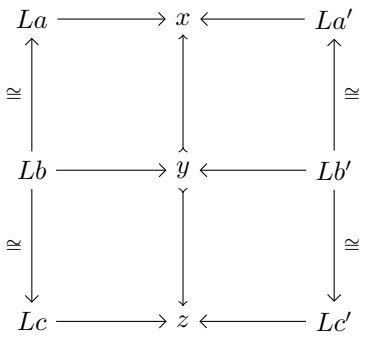
□

With horizontal and vertical composition in hand, we construct the double category ${}_L\mathbb{R}\mathbf{ewrite}$. Actually, we delay discussing the interchange law until Section **??** because it is difficult enough to warrant its own section.

Proposition 5.4. *Let*

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

*be a adjunction with L preserving pullbacks. There is a double category ${}_L\mathbb{R}$***rewrite** *whose objects are the A-objects, horizontal arrows of type $a \to b$ are structured cospans $La \to x \leftarrow Lb$, vertical arrows of type $a \to b$ are isomorphism classes of spans $a \leftarrow c \to b$ in A with invertible arrows, and squares are rewrites of structured cospans*



Proof. This proof requires we check the axioms of a double category. For simplicity, we denote ${}_L\mathbb{R}$**rewrite** by $\mathbb{R}$ in this proof.

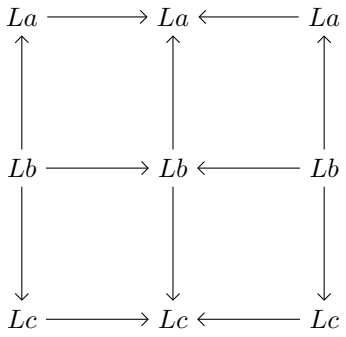
The object category $\mathbb{R}_0$ is given by objects of A and isomorphism classes of spans in A such that each leg is an isomorphism. The arrow category $\mathbb{R}_1$ has as objects the structured cospans

$$La \to x \leftarrow La'$$

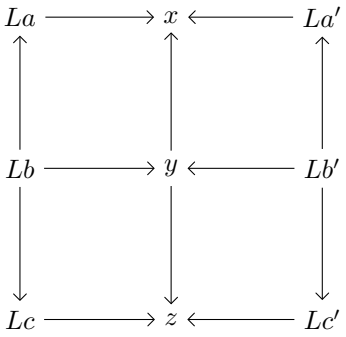
and as morphisms the fine rewrites of structured cospans.

The functor $U \colon \mathbb{R}_0 \to \mathbb{R}_1$ acts on objects by mapping a to the identity cospan on La and on morphisms by mapping $La \leftarrow Lb \to Lc$, whose legs are isomorphisms,

to the square



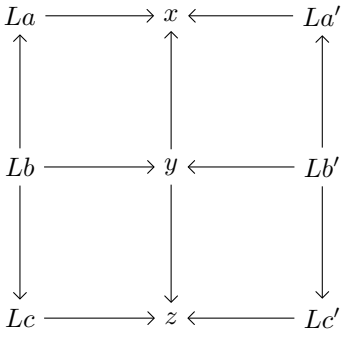
The functor $S\colon \mathbb{R}_1 \to \mathbb{R}_0$ acts on objects by sending $La \to x \leftarrow La'$ to a and on morphisms by sending a square



to the span $La \leftarrow Lb \to Lc$. The functor T is defined similarly sends an object

$$La \to x \leftarrow La'$$

of $\mathbb{R}_1$ to a' a square

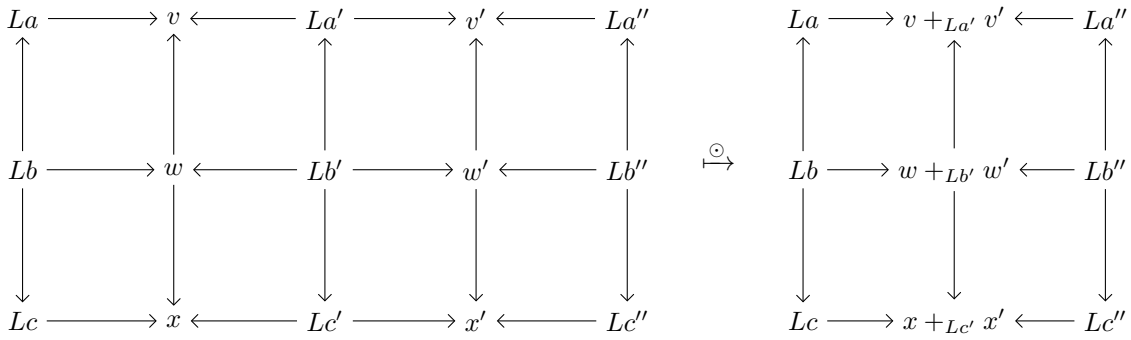


to the span $La' \leftarrow Lb' \rightarrow Lc'$.

The horizontal composition functor

$$\odot \colon \mathbb{R}_1 \times_{\mathbb{R}_0} \mathbb{R}_1 \to \mathbb{R}_1$$

acts on objects by composing cospans with pushouts in the usual way. It acts on morphisms by



The interchange law holds [2, Lem. 4.3], therefore $\odot$ is a functor. It is straightforward to check that the required equations are satisfied. The associator and unitors are given by natural isomorphisms that arise from universal properties. □

And now, our double category of rewrites is defined.

6. Expressiveness of grammars

For a rewrite rule $\ell \leftarrow k \rightarrow r$, there is no constraint on the value of k. Even though we interpret it as being the suboject of ℓ and r that remains fixed through the application of the rule, it can actually be any object of the topos in which we are working. Such freedom is difficult to work with, so we devise a more restrained alternative. That is, we ask that k be *discrete*. In this section, we make the term 'discrete' precise and show that the rewriting relation of a grammar (T, P) is the

same as the discretized variant of (T, P). This result generalizes a characterization of discrete graph grammars given by Ehrig, et. al. [3, 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 [5, Ch. C3.6]). The cost of avoiding this detour is one comonad.

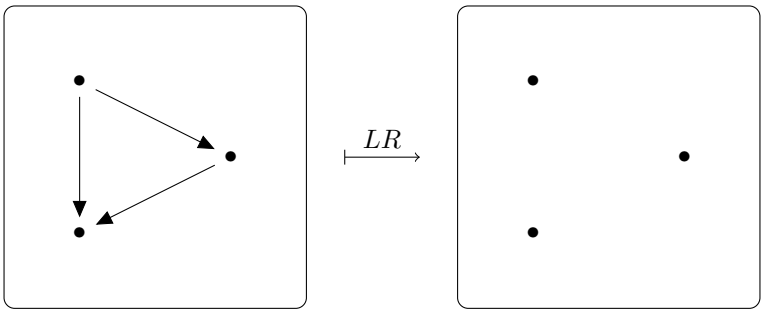
The definition of the 'discrete comonad' is straightforward enough, but its purpose may seem alien at first. After the definition, we explain its role in rewriting structured cospans.

Definition 6.1 (Discrete comonad)**.** A comonad on a topos is called **discrete** if its counit is monic. We use $\flat$ to denote a discrete comonad.

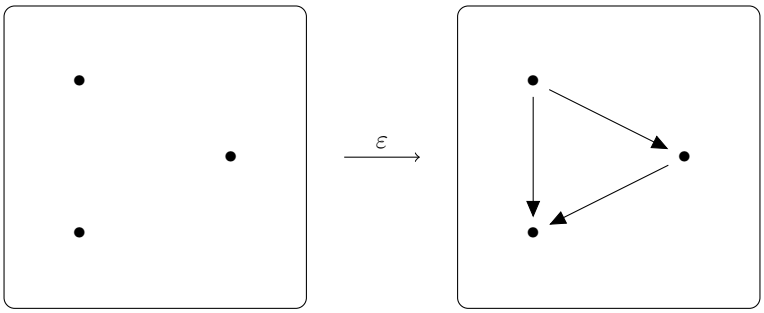
Secretly, we have been working with a discrete comonad all along. The adjunction

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

induces the comonad LR on RGraph. Applying LR to a graph x returns the edgeless graph underlying x, hence the term 'discrete'. For example



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,



Abstractly, this inclusion is why we ask for the counit to be monic. The property we capture with a discrete comonad comes from the systems interpretation of the adjunctions

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

between topoi. That is, R takes a system x, identifies the largest sub-system that can serve as an interface and turns that sub-system into an interface type Rx. Then L takes that interface type and turns it back into a system LRx. This process effectively strips away every part of a system leaving only those parts that can connect to the outside world. That means LRx is a part of x or, in the parlance of category theory, LRx is a subobject of x. Hence, we ask for a monic counit.

How do we plan to use discrete comonads? 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 6.2 (Discrete grammar)**.** Given a grammar (X, P), define $P_\flat$ as the set consisting of a rule

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

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

Theorem 6.10 gives a sufficient condition that allows us to swap (X, P) for $(\mathsf{X}, P_\flat)$ without consequence. The proof of this theorem borrows from lattice theory, so let us digress from rewriting to fill in the requisit background.

Definition 6.3 (Lattice)**.** A lattice is a poset $(S, \leq)$ equipped with all finite joins $\bigvee$ and all finite meets $\bigwedge$. It follows that there is a minimal 0 element and maximal element 1, realized as the empty meet and join, respectively.

Joins and meets are also known as suprema and infima. We are using the definition of a lattice common in the category theory literature. This leaves out objects that some mathematicians might consider lattices. Below we give one counterexample and several examples of lattices, the last one being the most relevant.

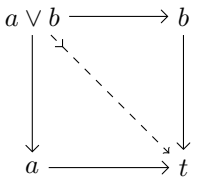
Example 6.4 (Integer Lattice)**.** The integers with the usual ordering $\leq$ do not form a lattice because there is no minimal or maximal element.

Example 6.5 (Lattice of power sets)**.** For any set S, its powerset $\mathcal{P}S$ is a poset via subset inclusion. The powerset becomes a lattice by taking join to be union $a \vee b \coloneqq a \cup b$, and meet to be intersection $a \wedge b \coloneqq a \cap b$. In general, union and intersection are defined over arbitrary sets, thus realizing arbitrary joins $\bigvee a_\alpha$ and arbitrary meets $\bigwedge a_\alpha$.

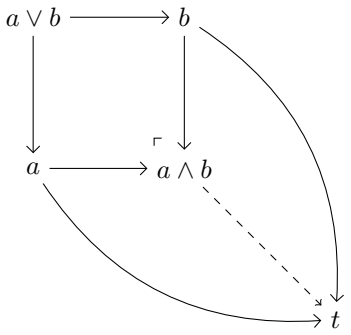
Those few examples provide intuition about lattices, but the next example is the most important lattice for us. It is the mechanism by which the power set is generalized into topos theory. It is called the subobject lattice.

Example 6.6 (Subobject lattice)**.** Let T be a topos and t be an object. There is a lattice $\mathrm{Sub}(t)$ called the subobject lattice of t. The elements of $\mathrm{Sub}(t)$, called subobjects, are isomorphism classes of monomorphisms into t. Here, two monomorphisms f, g into t are isomorphic if there is a commuting diagram

The order on $\mathrm{Sub}(t)$ is given by $f \leq g$ if f factors through g, meaning there is an arrow $h\colon a \to b$ such that $f = gh$. Note that h is necessarily monic. The meet operation in $\mathrm{Sub}(t)$ is given by pullback



and join is given by pushout over the meet



Subobject lattices provide a way to compare the expressivity of a grammar to its underlying discrete grammar. To do this, we require a subobject lattice with arbitrary meets. The powerset lattice mentioned above has this property, but when does a subobject lattice have this property? Here are several sufficient conditions, starting with a well-known result from order theory.

Proposition 6.7. *Any lattice that has all joins also has all meets.*

Proof. Consider a subset S of a lattice. Define the meet of S to be the join of the set of all lower bounds of S. □

Proposition 6.8. *Fix an object t of a topos T. The subobject lattice* $\mathrm{Sub}(t)$ *has arbitrary meets when the over-category $T \downarrow t$ has all products.*

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

Corollary 6.9. *Fix an object t of a topos T. The subobject lattice* $\mathrm{Sub}(t)$ *has arbitrary meets when the over category $T \downarrow t$ has all coproducts.*

Proof. Combine Propositions 6.7 and 6.8. □

The case where T is a presheave category is especially pertinent to us because, often, a system can be encoded as a presheaf. Labelled graphs offer a large cache of examples.

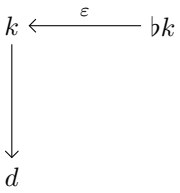
At last, we combine the discrete comonad, the discrete grammar, and the complete subobject lattice into a result on the expressiveness on discrete grammars.

Theorem 6.10. *Let T be a topos and $\flat\colon \mathsf{T} \to \mathsf{T}$ be a discrete comonad. Let (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. Then 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

(6)
$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \downarrow & \lrcorner & \downarrow & \llcorner & \downarrow \\ g & \longleftarrow & d & \longrightarrow & h \end{array}$$

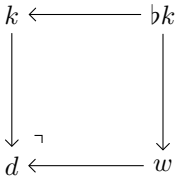
we can achieve that same derivation using rules in $P_\flat$. This requires we build a pushout complement w of the diagram



Define

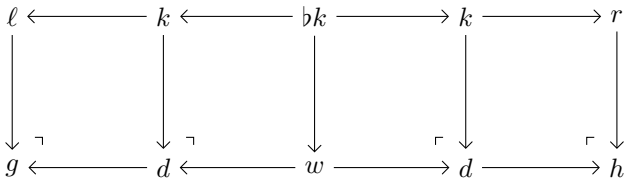
$$w \coloneqq \bigwedge \{z \colon z \vee k = d\} \vee \flat k,$$

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



is a pushout. It follows that there is a derivation

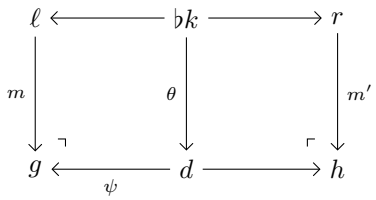
(7)



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

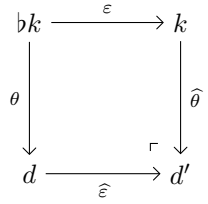
(8)



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

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

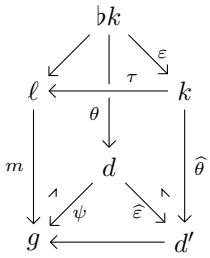
in P, meaning that the map $\flat k \to \ell$ factors through τ. Next, define d' to be the pushout of the diagram



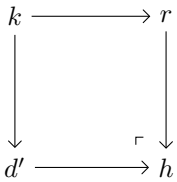
By invoking the universal property of this pushout with the maps

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

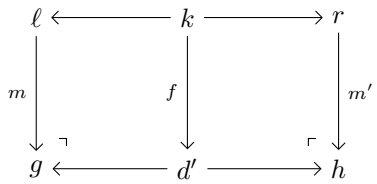
we get a canonical map $d' \to g$ that we can fit into a commuting diagram



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



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. □

7. Characterizing the rewrite relation for closed systems

Here, we apply the theory of rewriting structured cospans to classifying the rewrite relation for closed systems. We start by formalizing closed systems as structured cospans with an empty interface $L0 \to x \leftarrow L0$. In other words, closed systems are particular horizontal 1-arrows in the double category ${}_L\mathbb{R}\mathbf{ewrite}$. 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 structured cospans

$$L0 \to x \leftarrow L0 \quad \text{and} \quad L0 \to x' \leftarrow L0$$

are equivalent precisely when there is a square between them. We interpret this result in three ways.

(a) It shows that the rewriting relation for a closed system is functorial and is characterized using squares in a double category.

(b) A closed system decomposes into open systems and simplifying each open system simplifies the composite closed system.

(c) Open systems provide a local perspective on closed systems via this decomposition.

To decompose closed systems into open systems, we need a topos of closed systems X equipped with a grammar (X, P). The closed systems need interfaces, meaning we need to introduce an adjunction

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

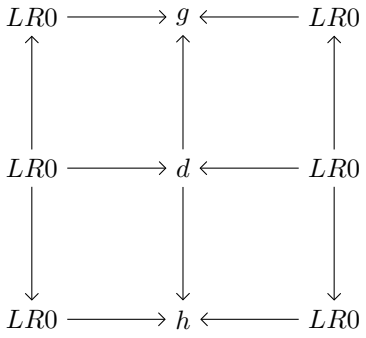
where L preserves pullbacks and has a monic counit. At this point, the material from the previous section returns. This adjunction gives a discrete comonad $\flat := LR$ from which we form the discrete grammar $(\mathsf{X}, P_\flat)$. Now define the structured cospan grammar $({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ where $\widehat{P_\flat}$ contains the rule

$$\begin{array}{ccccc} L0 & \longrightarrow & \ell & \longleftarrow & LRk \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longrightarrow & LRk & \longleftarrow & LRk \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longrightarrow & r & \longleftarrow & LRk \end{array} \tag{9}$$

for each rule $\ell \leftarrow LRk \rightarrow r$ of $P_\flat$. We use $({}_L\mathsf{StrCsp}, \widehat{P_\flat})$ to prove our main theorem.

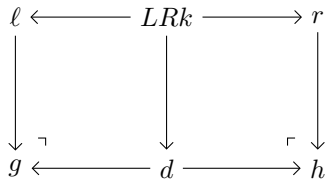
Before stating the theorem, we note that this theorem generalizes work by Gadducci and Heckel [4] whose domain of inquiry was graph rewriting. The arc of our proof follows theirs.

Theorem 7.1. *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*

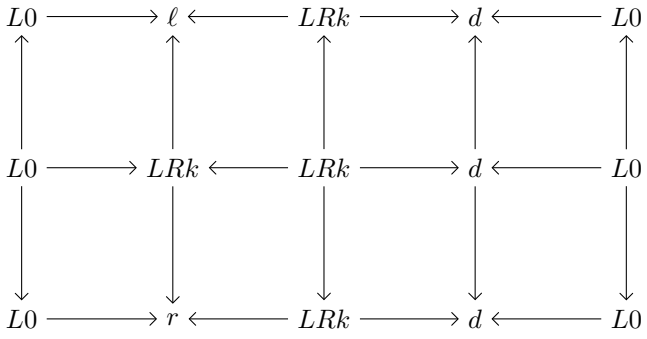


in the double category $\mathrm{Lang}({}_L\mathsf{StrCsp}, \widehat{P_\flat})$.

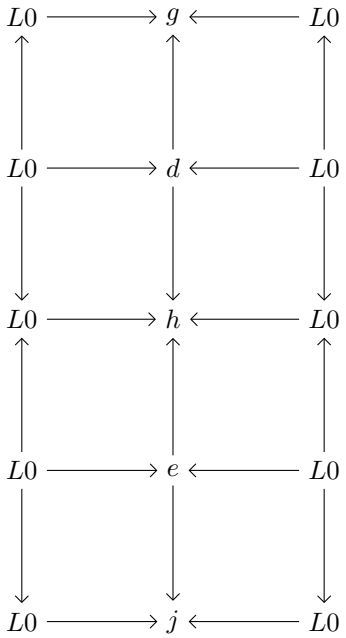
Proof. We show sufficiency by inducting on the length of the derivation. If $g \rightsquigarrow^* h$ in a single step, meaning that there is a diagram



then the desired square is the horizontal composition of

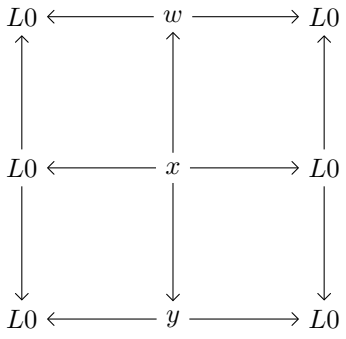


The left square is a generator and the right square is the identity on the horizontal arrow $LRk \to d \leftarrow L0$. The square for a derivation $g \rightsquigarrow^* h \rightsquigarrow j$ is the vertical composition of



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

$$\begin{array}{ccccc} L0 & \longleftarrow & y & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & L0 & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & z & \longrightarrow & L0 \end{array}$$

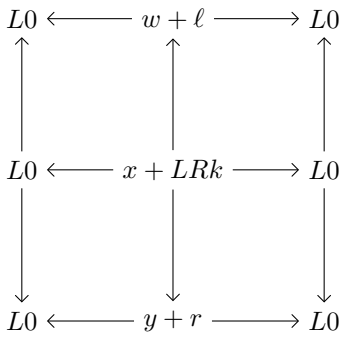
corresponding to a production $y \leftarrow L0 \rightarrow z$ gives

$$\begin{array}{ccccc} L0 & \longleftarrow & w & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & L0 & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & z & \longrightarrow & L0 \end{array}$$

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

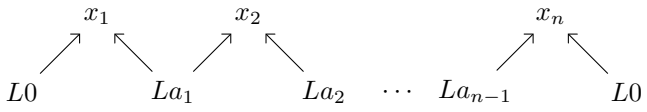
$$\begin{array}{ccccc} L0 & \longleftarrow & \ell & \longrightarrow & L0 \\ \uparrow & & \uparrow & & \uparrow \\ L0 & \longleftarrow & LRk & \longrightarrow & L0 \\ \downarrow & & \downarrow & & \downarrow \\ L0 & \longleftarrow & r & \longrightarrow & L0 \end{array}$$

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

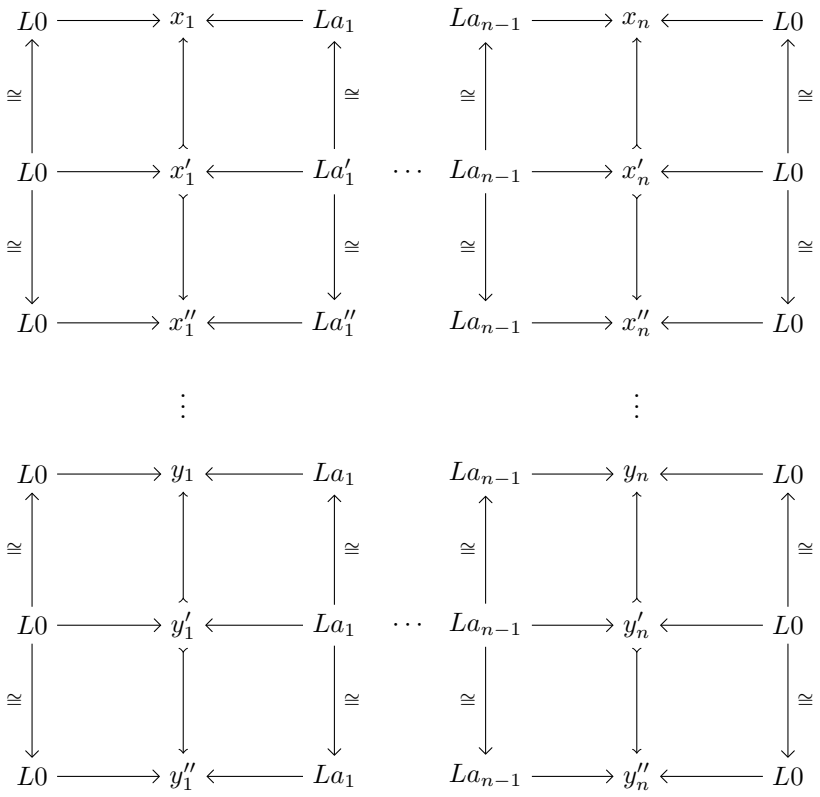


But $w + \ell \rightsquigarrow^* y + r$ as seen in Lemma 4.4. □

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



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 C. Baez and Kenny Courser. Structured cospans. *In Preperation.* (Referred to on page 5.)

[2] 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 23.)

[3] 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 24.)

[4] 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 33.)

[5] Peter T Johnstone. *Sketches of an elephant: A topos theory compendium*, volume 2. Oxford University Press, 2002. (Referred to on page 24.)

[6] 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 1, 2, 13, 15.)

[7] Gavin Wraith. Artin glueing. *J. Pure Appl. Algebra*, 4:345–348, 1974. (Referred to on page 8.)

Email address: dcicala@newhaven.edu

Department of Mathematics and Physics, University of New Haven