Mathematical Structures in Computer Science (2026), **36**, e28, pp. 1–19
doi:10.1017/S0960129526100668

CAMBRIDGE UNIVERSITY PRESS

PAPER

Rewriting structured cospans

Daniel Cicala

Mathematics, Southern Connecticut State University, USA
Email: cicalad1@southernct.edu

(Received 5 September 2025 UTC; revised 2 April 2026 UTC; accepted 9 August 2026 UTC)

Abstract

We develop a theory of rewriting for structured cospans in order to extend compositional methods for modeling open networks. First, we introduce a category whose objects are structured cospans and establish conditions under which it is adhesive or a topos. These results guarantee that double pushout rewriting can be applied in this setting. We then define structured cospan grammars and construct their associated languages via a 2-categorical framework, capturing both network composition and rewrite dynamics. As an application, we show that for graphs, hypergraphs, Petri nets, and their typed variants, any grammar induces the same language as its corresponding discrete grammar. This equivalence enables an inductive characterization of rewriting, thereby generalizing classical results from graph transformation to a broader class of categorical models.

Keywords: Structured cospans; double pushout rewriting; open networks; compositionality

1. Introduction

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

Baez and Courser introduced structured cospans as an abstract framework to model open networks (Baez and Courser 2020; Courser 2020). A structured cospan is a diagram of the form

$$La \to x \leftarrow Lb \tag{1}$$

where $L\colon \mathsf{A} \to \mathsf{X}$ is a functor whose codomain has chosen pushouts. This functor is a convenient bookkeeping device allowing us to separate open networks into the interface types A and the network types X. We interpret (1) as a network x with inputs La and outputs Lb. This network can then *connect* to a compatible network, that is one with inputs Lb, by pushout:

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

We then get a category ${}_L\mathsf{Csp}$ whose objects are those of A and whose arrows are structured cospans $La \to x \leftarrow Lb$ where composition models the connecting of networks.

Given the ability to model open networks using structured cospans, we would like tools to analyze these models. In this paper, we adapt for structured cospans the tool of double pushout (DPO) rewriting, an algorithmic technique for creating a new object from an old object according to a given set of rules.

© The Author(s), 2026. Published by Cambridge University Press.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

1.1 Double pushout rewriting

Rewriting offers a method of replacement, often for the purpose of generating or simplifying. There are numerous examples. In the case of electrical circuits, rewriting provides a method to replace two resistors in series with a single resistor. In database schemas, rewriting offers a way to transform one schema into another (Rodgers and King 1997). Rewriting generates strings in formal languages (Baader and Nipkow 1998).

DPO rewriting is an algebraic mechanism that can perform the rewriting. It works in any adhesive category (Lack and Sobociński 2004). One starts with a *grammar* (C, P), that is an adhesive category C and a set $P := \{\ell_j \leftarrow k_j \rightarrow r_j\}$ of spans in C with monic arrows. Each span is called a *rewrite rule*, or just *rule* for short. We interpret a rule $\ell \leftarrow k \rightarrow r$ as stating that r replaces ℓ in a manner that fixes k. This rule can be applied to any object ℓ' by realizing a *DPO diagram*.

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ {\scriptstyle m}\downarrow & \lrcorner & \downarrow & \lrcorner & \downarrow \\ \ell' & \longleftarrow & k' & \longrightarrow & r' \end{array} \tag{2}$$

where m identifies an instance of ℓ in ℓ' and replaces it with r. The resulting object of this rewriting process is r'. The squares being pushouts means that the *gluing condition* is met, which ensures the process outputs a well-defined object. For instance, in the case of graphs, the gluing condition means that there no node is removed leaving an edge unanchored. By considering all possible DPO diagrams (2) in C where the top row is in P, we get a relation $\ell' \rightsquigarrow r'$ on the objects of C. The primary item to study is the *rewrite relation* $\rightsquigarrow^*$ which is the reflexive and transitive closure of $\rightsquigarrow$. One may alternatively study the *language* of a grammar, which is the category whose objects are those of C and arrows are generated by the rewrite relation. Both the rewrite relation and the language encode all the possible ways to rewrite one object into another.

1.2 Rewriting structured cospans

Observe that rewriting applies to the *objects* of an adhesive category. So instead of working with the category ${}_L\mathsf{Csp}$ where structured cospans are arrows, we need to construct a category where they are objects. We denote this by ${}_L\mathsf{StrCsp}$. To build it from scratch, start with an adjunction $L \dashv R \colon X \to A$. We may consider that X contains every network type, A every interface type, L includes each interface type into X as a sort of trivial network type, and R returns what may loosely be considered the largest possible interface of a network. Now, define ${}_L\mathsf{StrCsp}$ to have structured cospans for objects and commuting diagrams

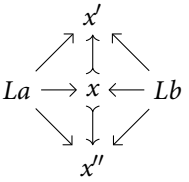
$$\begin{array}{ccccc} La & \longrightarrow & x & \longleftarrow & Lb \\ {\scriptstyle Lf}\downarrow & & {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle Lh} \\ La' & \longrightarrow & x' & \longleftarrow & Lb' \end{array}$$

as arrows. We show that if X and A are adhesive and L preserves pullbacks, then ${}_L\mathsf{StrCsp}$ is adhesive (Theorem 5). Also, if we strengthen this to make $L \dashv R$ a geometric morphism between topoi, then ${}_L\mathsf{StrCsp}$ is a topos (Theorem 6). It follows that, under such modest conditions, ${}_L\mathsf{StrCsp}$ supports rewriting.

Though we restrict L more than Baez and Courser, our definition still covers many important examples. One such example is the discrete graph functor $L \colon \mathsf{Set} \to \mathsf{Graph}$. More examples come by using slice categories Graph/g for some graph g chosen to endow nodes and arrows with types. This was done to model the ZX-calculus (Cicala 2018a) and can be done to model passive linear circuits (Baez and Fong 2018) by choosing g to have a single node and edge set of possible resistances $(0, \infty)$. Other examples include directed hypergraphs, undirected hypergraphs, their typed versions, Petri nets, and marked Petri nets.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

To rewrite structured cospans, we begin with a *structured cospan grammar* $({}_L\mathsf{StrCsp}, P)$, which is different from a grammar as discussed above because we require P to contain spans in ${}_L\mathsf{StrCsp}$ of the form



which rewrites $La \to x' \leftarrow Lb$ into $La \to x'' \leftarrow Lb$. This condition is stronger than simply requiring monic-legged spans because the interfaces are fixed.

Earlier, we associated to a grammar a 1-category called the "language". Now, with a structured cospan grammar $({}_L\mathsf{StrCsp}, P)$, we construct its language $\mathbf{Lang}({}_L\mathsf{StrCsp}, P)$ as a free 2-category on a computad (Street 1976). The reason for a 2-category is that there are two compositional structures at play. There is the composing of structured cospans and of the rewrite rules. These will respectively form the 1- and 2-morphisms of $\mathbf{Lang}({}_L\mathsf{StrCsp}, P)$.

1.3 An application: An inductive viewpoint on rewriting

In the classical topics of rewriting – formal languages and term rewriting – there are two approaches to defining the rewrite relation for a grammar. The first is an operational definition that stipulates when a rule can be applied by using sub-terms and substitution. The second is an inductive definition that constructs the rewrite relation using generators and closure operations. When rewriting theory expanded to graphs in the 1970s, only the operational definition prevailed. The DPO mechanism performed the substitution. Then in the 1998, Gadducci and Heckel introduced an inductive definition to graph rewriting (Gadducci and Heckel 1998), thus allowing for analyses using structural induction. With the new technology of structured cospans, we can use their ideas to bring the inductive viewpoint to rewriting in a broader class of adhesive categories beyond directed graphs.

A first key idea in developing the inductive definition of graph rewriting was to use open graphs. To *openify* objects beyond graphs, we use structured cospans.

A second key idea is using an equivalence between two classes of rewrite relations. In the context of graph rewriting, this result states that the rewrite relation for a graph grammar $(\mathsf{Graph}, \{\ell_j \leftarrow k_j \to r_j\})$ is the same as for its underlying *discrete graph grammar* $(\mathsf{Graph}, \{\ell_j \leftarrow \flat k_j \to r_j\})$ where we replace the graphs k_j with their underlying discrete graphs $\flat k_j$ and restrict the maps accordingly. To extend this to rewriting in adhesive categories, we start with an adjunction $L \dashv R$ with a monic counit and interpret the comonad $\flat := LR$ as sending an object x to the *discrete* object $\flat x$ underlying x.

In summation, we have the necessary ingredients to pursue an inductive viewpoint of rewriting: structured cospans provide objects with interfaces, and the comonad $\flat$ provides discrete grammars. With these, we generalize Gadducci and Heckel's inductive-style rewriting on graphs to a broader collection of objects (Theorem 27), including hypergraphs and Petri nets.

1.4 Outline and contributions

Section 2 contains a brief overview of background material on DPO rewriting and structured cospans.

Section 3 introduces rewriting to structured cospans. Section 3.1 contains the main result of this paper; after introducing a new category ${}_L\mathsf{StrCsp}$ with structured cospans as objects, we show that this category is adhesive (Theorem 5) and, under slightly stronger conditions, a topos (Theorem 6)

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

constructed functorially in L (Theorem 7). After discussing structured cospan grammars and their languages in Section 3.2, we briefly mention the properties that they then inherit in Section 3.3.

Section 4 uses structured cospans to bring an inductive viewpoint of rewriting to a broader class of objects. A brief history of the inductive viewpoint is provided in Section 4.1. Discrete grammars are introduced in Section 4.2 followed, in Section 4.3, by a sufficient condition for a grammar and its discrete grammar to induce the same language (Lemma 15). This condition is shown to hold for directed hypergraphs, undirected hypergraphs, Petri nets, marked Petri nets, and typed objects as constructed using slice categories. Finally, the main application of this paper is given in Section 4.4: the construction of the inductive viewpoint (Theorem 27).

1.5 Related work

The theory of rewriting is a mature field, particularly the rewriting of various kinds of graphs and the axiomatization of graph rewriting via adhesive categories.

The concept of open networks has been previously explored, including open graphs, open hypergraphs, and open Petri nets (Baldan et al. 2001; Sassone and Sobociński 2004; Dixon and Kissinger 2013). Approaches included various techniques to equip graph-type objects with an interface. Structured cospans provided a unified technique that is applicable to a broader class of open networks (Cicala 2018b; Baez and Courser 2020; Baez and Master 2020; Baez et al. 2022). An alternative approach is Fong's decorated cospans (Fong 2015), which we do not address here. Rewriting open graphs, hypergraphs, Petri nets, and string diagrams have been studied (Bonchi et al. 2016, 2017; Ehrig and König 2004; Sassone and Sobociński 2004; Sassone and Sobociński 2005), but this paper is the first to introduce rewriting specifically to structured cospans. This contribution places previous efforts into a unified framework, using general categorical techniques.

In addition to the above, this paper draws on the theory of adhesive categories (Lack and Sobociński 2006) and the presheaf perspective on both hypergraphs (Bonchi et al. 2017) and Petri nets (Johnstone et al. 2007).

2. Background Material: DPO Rewriting and Structured Cospans

This section contains background material on our main topics: DPO rewriting and structured cospans. Nothing novel is presented here.

2.1 Double pushout rewriting

DPO rewriting has an established literature, so we use this section to cover the fundamentals and to establish our conventions. The interested reader can see Ehrig et al. (1973) to learn about graph rewriting or Lack and Sobociński (2004) for an axiomatic approach based on adhesive categories.

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. A **grammar** (C, P) is defined to be an adhesive category C together with a finite set of rules $P := \{\ell_j \leftarrow k_j \rightarrow r_j\}$.

We can apply a rule $\ell \leftarrow k \rightarrow r$ from P to an object ℓ' of C using any arrow $m\colon \ell \rightarrow \ell'$ for which there exists a pushout complement, that is, an object k' fitting into a pushout diagram

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

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

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 (Lack and Sobociński 2004, Lemma 15).

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

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

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

A grammar (C, P) induces a collection dP of all derived rules obtained by applying a rule in P to an object in C. We can use dP to analyze the grammar (C, P) by constructing the *rewrite relation* $\rightsquigarrow^*$. The meaning of $x \rightsquigarrow^* y$ is that we can rewrite x into y by applying a sequence of rules in dP. To precisely define the rewrite relation, we start by constructing a relation $\rightsquigarrow$ on the objects of C by setting $\ell' \rightsquigarrow r'$ if there exists a rule $\ell' \leftarrow k' \to r'$ in dP. However, $\rightsquigarrow$ does not capture enough information about (C, 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, which we call the **language of a grammar** $\mathrm{Lang}(\mathsf{C}, P)$, whose objects are those of C and arrows are generated by $\rightsquigarrow$. The language is constructed so that there is an arrow $x \to y$ in $\mathrm{Lang}(\mathsf{C}, P)$ if and only if $x \rightsquigarrow^* y$.

In the rewriting literature, the terms "language" and "rewrite relation" are often interchangeable. However, we give them slightly different meanings in order to help orient the reader. Namely, we use "rewrite relation" when giving a relational perspective and "language" when giving the category theoretical perspective.

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

2.2 Structured cospans

Baez and Courser (2020) introduced structured cospans as a framework to study open networks. A network is **open** when equipped with a mechanism by which it can connect to any compatible network. For example, a vacuum cleaner can connect with the electrical grid via an electrical socket. A pulley network can connect to a mechanical motor. An open network stands in contrast to a closed network that cannot interact with its outside environment.

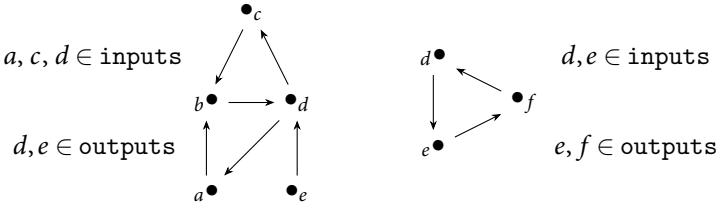
Definition 1 (Structured cospans). *Given a functor $L\colon \mathsf{A} \to \mathsf{X}$ where X has pushouts, an L-**structured cospan**, or simply **structured cospan**, is a diagram $La \to x \leftarrow Lb$ in X.*

To interpret $La \to x \leftarrow Lb$ as an open network, take x to represent the network with inputs La chosen by the arrow $La \to x$ and outputs Lb chosen by $x \leftarrow Lb$. This open network can now connect to any other open network with inputs Lb, say $Lb \to y \leftarrow Lc$. We form the *composite* of the two open networks 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 network $La \to x +_{Lb} y \leftarrow Lc$. We capture this in a category ${}_L\mathsf{Csp}$ whose objects are those of A and arrows $a \to b$ are structured cospans $La \to x \leftarrow Lb$ with composition given by pushout.

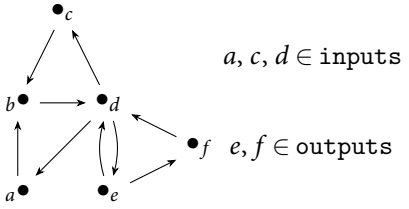
Example 2. Structured cospans can be used to slightly generalize open graphs. Set theoretically, a graph is open when equipped with two subsets of its nodes, one set serving as inputs and the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

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



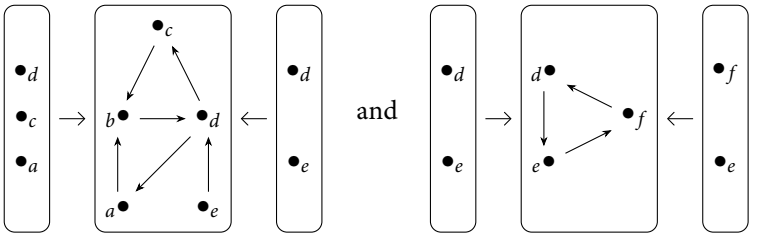
compose by gluing the corresponding nodes together, forming the new open graph



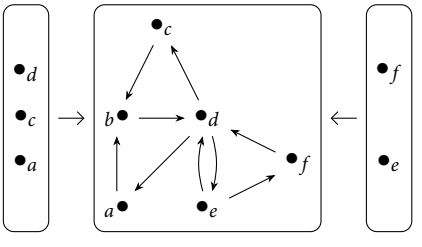
To define an open graph as a structured cospan, consider

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

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



with composite



Note, that the functions forming the legs of the cospan need not be monic, hence calling the structured cospan version of open graphs a slight generalization of the set theoretic version of open graphs. In this example, the category ${}_L\mathsf{Csp}$ has sets for objects and open graphs for arrows.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

3. Structured Cospans and Rewriting

In this section, we give our main result that structured cospans form an adhesive category or a topos under mild conditions (Section 3.1). It follows from this result that structured cospans admit a rewriting theory. We then define a grammar and its language for structured cospans (Section 3.2). These require a bit more discussion than simply importing the ideas from the current rewriting literature. Finally, we reflect on some of the properties that rewriting structured cospans will inherit, simply as a matter of being adhesive (Section 3.3).

3.1 Characterizing structured cospan categories as adhesive or topoi

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

Recall that, in Section 2, we saw that rewriting operates on the objects of an adhesive category, not the arrows. Therefore, we cannot hope to rewrite structured cospans inside the category ${}_L\mathsf{Csp}$. Our task, now, is to build a category where structured cospans are objects and then to show that it is adhesive.

Definition 3. *Fix an adjunction $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. Define ${}_L\mathsf{StrCsp}$ to be the category whose objects are structured cospans and arrows from $La \to x \leftarrow Lb$ to $La' \to x' \leftarrow Lb'$ are triples of arrows (f, g, h) fitting into the commuting diagram*

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

Now, in order to enable the rewriting of structured cospans in the sense of Section 2, ${}_L\mathsf{StrCsp}$ must be adhesive. To achieve this, we outline two sets of conditions in order to later prove results of different strengths.

Definition 4. *Fix an adjunction $L \dashv R \colon \mathsf{X} \to \mathsf{A}$. We say that this adjunction satisfies the **adhesivity condition** if X and A are adhesive and L preserves pullbacks. We say that this adjunctions satisfies the **topos condition** if R is a geometric morphism between topoi.*

Every topos is adhesive (Lack and Sobociński 2006), meaning the topos condition is strictly stronger than the adhesive condition. As shown in the next two results, when structured cospans are built with $L \dashv R$ satisfying the adhesivity condition, then ${}_L\mathsf{StrCsp}$ is adhesive. Similarly, ${}_L\mathsf{StrCsp}$ is a topos when $L \dashv R$ satisfies the topos condition.

Theorem 5. *If $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ satisfies the adhesive condition, then ${}_L\mathsf{StrCsp}$ is adhesive.*

Proof. Pullbacks exist in ${}_L\mathsf{StrCsp}$ because both X and A have pullbacks and L preserves them. The same is true about pushouts of monos. That a pushout of a monic map is a Van Kampen square follows from this property holding in both X and A and L preserving pullbacks and pushouts. □

While in this paper, we will largely work with the generality of adhesive categories, many objects of rewriting interest actually form topoi. Graphs are the primary example. Therefore, we show that under slightly stronger conditions, structured cospans form a topos and that this construction is functorial in L.

Theorem 6. *If $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ satisfies the topos condition, then ${}_L\mathsf{StrCsp}$ is a topos.*

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

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

$$\begin{array}{ccccc} a & \longrightarrow & Rx & \longleftarrow & b \\ {\scriptstyle f}\downarrow & & {\scriptstyle Rg}\downarrow & & \downarrow{\scriptstyle h} \\ a' & \longrightarrow & Rx' & \longleftarrow & b' \end{array}$$

This, in turn, is equivalent to the comma category $(\mathsf{A} \times \mathsf{A}) \downarrow \Delta R$ where $\Delta \colon \mathsf{A} \to \mathsf{A} \times \mathsf{A}$ is the diagonal functor, a right adjoint. Because ΔR is a right adjoint, $(\mathsf{A} \times \mathsf{A}) \downarrow \Delta R$ is an Artin gluing (Wraith 1974), therefore a topos. □

Theorem 7. *Denote by* Topos *the category of topoi and geometric morphisms. There is a contravariant functor* ${}_{(-)}\mathsf{StrCsp} \colon [\bullet \to \bullet, \mathsf{Topos}] \to \mathsf{Topos}$ *defined by*

$$\begin{array}{ccc} \mathsf{X} & \overset{L}{\underset{R}{\leftrightarrows}} & \mathsf{A} \\ F \dashv G\uparrow\downarrow & & F' \dashv G'\uparrow\downarrow \\ \mathsf{X}' & \underset{R'}{\overset{L'}{\leftrightarrows}} & \mathsf{A}' \end{array} \quad \overset{{}_{(-)}\mathsf{StrCsp}}{\longmapsto} \quad {}_{L'}\mathsf{StrCsp} \underset{\Psi}{\overset{\Theta}{\leftrightarrows}} {}_{L}\mathsf{StrCsp}$$

which is in turn given by

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

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

Proof. In light of Theorem 6, it suffices to show that $\Theta \dashv \Psi$ gives a geometric morphism. Let ℓ and ℓ', respectively, denote the L-structured cospan $La \to x \leftarrow Lb$ and 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

$$\begin{aligned} \big((f, g, h) \colon \ell \to \Psi\ell'\big) &\mapsto \big((\varepsilon' \circ G'f, \varepsilon \circ Gg, \varepsilon' \circ G'h) \colon \Theta\ell \to \ell'\big) \\ \big((f', g', h') \colon \Theta\ell \to \ell'\big) &\mapsto \big((Ff' \circ \eta', F'g' \circ \eta, F'h' \circ \eta') \colon \ell \to \Theta'\ell'\big) \end{aligned}$$

give a bijection $\hom(\Theta\ell, \ell') \simeq \hom(\ell, \Psi\ell')$. The naturality in ℓ and ℓ' follows from the naturality of η, ε, η', and ε'. The left adjoint Θ preserves finite limits because they are taken pointwise and L', G, and G' all preserve finite limits. Identity and composition are easily checked to be preserved. □

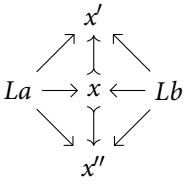
For the remainder of the paper, we work with structured cospans that satisfy the adhesive condition unless otherwise stated.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

3.2 Rewriting structured cospans

We now know that the category ${}_L\mathsf{StrCsp}$ of structured cospans and their morphisms are adhesive and, therefore, supports a rich rewriting theory. In this section, we begin to develop this theory.

Definition 8. *A **rewrite rule of structured cospans** is an isomorphism class of spans of structured cospans (Cicala and Courser 2018; Cicala 2018b) of the form*



The marked arrows $\rightarrowtail$ are monic.

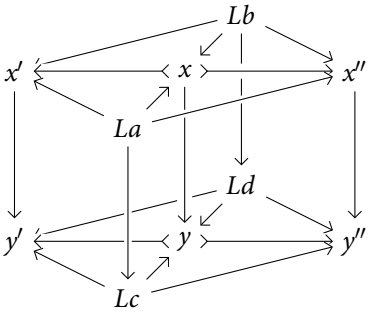
The conceit of this rule is that the structured cospan $La \to x' \leftarrow Lb$ along the top of the diagram is rewritten as the structured cospan $La \to x'' \leftarrow Lb$ along the bottom. Here, we are orienting our diagrams as "top rewrites to bottom" instead of "left rewrites to right". A nonsuperficial difference is that this diagram is not merely a span in ${}_L\mathsf{StrCsp}$ with monic legs. We disallow the inputs and outputs from being rewritten.

In order to apply this rule to other graphs, we need the notions of grammars and derived rules for structured cospans.

Definition 9. *A **grammar of structured cospans** is a pair $({}_L\mathsf{StrCsp}, P)$, where P is a set of rewrite rules of structured cospans.*

Proposition 10. *Derived rules of structured cospan rewrites are structured cospan rewrite rules.*

Proof. Consider the diagram

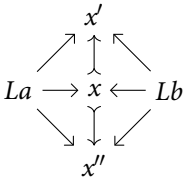


where the top face is a rewrite rule of structured cospans, and the back left and front left faces give a matching map, with pushout complement, of structured cospans. Then the bottom face is also a rewrite rule of structured cospans because pushouts preserve monos in an adhesive category. □

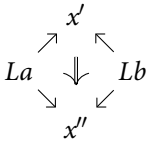
Now that we are certain that derivation preserves rewrite rules of structured cospans, we can construct the language. Similar to the construction in Gadducci and Heckel (1998), we realize the language as the free 2-category on a computad whose 0-cells are interface types, 1-cells are structured cospans, and 2-cells are generated by the rewrite rules of a given grammar.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

Definition 11. *Given an adjunction* $L \dashv R \colon X \to A$ *satisfying the adhesivity condition (Definition 4) and a structured cospan grammar* $({}_L\mathsf{StrCsp}, P)$*, define a computad* $\mathrm{Comp}({}_L\mathsf{StrCsp}, P)$ *with* 0*-cells the* A*-objects,* 1*-cells the arrows of* ${}_L\mathsf{StrCsp}$ *and, for each*



derived from P*, a* 2*-cell* γ_p



Apply to this computad the free 2-category functor, that is the left adjoint to the forgetful functor taking each 2-category to its underlying computad. The resulting 2-category is what we call $\mathbf{Lang}({}_L\mathsf{StrCsp}, P)$.

Note that we have two notions of language. The first, written in non-bold type, $\mathrm{Lang}(\mathsf{X}, P)$ is a 1-category generated by the rewrite relation of a typical grammar. The second, written in bold type, $\mathbf{Lang}({}_L\mathsf{StrCsp}, Q)$ is the 2-category generated by a computad for a structured cospan grammar.

3.3 Basic properties inherited

Thus far, we have constructed an adhesive category ${}_L\mathsf{StrCsp}$. Therefore, it enjoys the basic properties of adhesive categories, as detailed by Lack and Sobociński (2004). Moreover, structured cospan grammars involve rewrite rules that are stronger than those for general adhesive grammars. For general adhesive categories, rewrite rules can be any span with monic legs, while structured cospan rewrite rules are *certain* spans with monic legs. Therefore, a structured cospan grammar is just a grammar in an adhesive category. It follows that the Local Church Rosser theorem and the Concurrency theorem hold for structured cospan grammars.

4. An Application: An Inductive Perspective of Rewriting

In this section, we shift our perspective from rewriting structured cospans to using structured cospans to provide an inductive viewpoint of rewriting. We start this section with a brief history of inductive rewriting, including its introduction to directed graphs. Then after establishing definitions, we close this section with sufficient conditions for categories to admit inductive rewriting on their objects. We illustrate this with directed and undirected hypergraphs.

4.1 A brief history of inductive rewriting

Before graph rewriting, there were both formal language rewriting and term rewriting. In these cases, there are two ways to define the rewrite relation. The first way is called the *operational method*, which applies a rule by substituting a sub-term for another term. The second way is called the *inductive method*, which constructs the rewrite relation using generators and closure operations. In classical graph rewriting, only the operational method existed, where substitution

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

was achieved with the DPO method. Eventually, Gadducci and Heckel introduced an inductive method to construct the rewrite relation, opening the way to analyze graph grammars through structural induction.

These authors accomplished this by defining what are now more commonly called *open graphs* defined using a cospan $d_1 \to g \leftarrow d_2$ where the images of discrete graphs d_1 and d_2 identified the inputs and outputs of a graph g. This idea is a precursor for structured cospans. The inductively defined rewrite relation for a graph grammar used this open graph construction to encode the relation inside the hom-set of a particular 2-category. We emphasize here that the goal was not about rewriting open graphs, but to use open graphs to talk about rewriting graphs.

In this section, we adapt their ideas to give an inductive definition of the rewrite relation for a grammar (X, P) such that X fits into an adjunction $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ satisfying the adhesivity condition (Definition 4) and has a monic counit. We also require the grammar to induce the same language as the *discrete grammar* obtained by replacing each rule $\ell \leftarrow k \to r$ in P with $\ell \leftarrow LRk \to r$. It is already known that the category of directed graphs satisfies these conditions (Ehrig et al. 1973, Proposition 3.3). We show this is also true of typed graphs, different flavors of hypergraphs, and Petri nets.

Before proceeding, we reemphasize that this section is not about rewriting structured cospans. Instead, we use structured cospans as a tool to inductively construct the language for a certain class of grammars.

4.2 Discrete grammars

In general, for any rewrite rule $\ell \leftarrow k \to r$, there is only one constraint on the value of k: it must be a subobject of ℓ and r. But requiring that k also be discrete can simplify any analysis involving that rule. This leads us question whether we can learn about a grammar (X, P) by instead studying the grammar $(\mathsf{X}, P_\flat)$, where $P_\flat$ is obtained by *discretizing* the apexes of every rule in P. In this section we explain the $\flat$ notation and make precise the concept of discreteness before giving the main result of this section that characterizes when (X, P) and $(\mathsf{X}, P_\flat)$ give the same language. This result generalizes the characterization of discrete graph grammars given by Ehrig et al. (1973, Proposition 3.3).

As a brief aside, experts in topos theory will know that discreteness comes from the flat modality on a local topos (Johnstone 2002, Ch. 3.6)). While this is not unrelated to our definition of discreteness, we are really borrowing this term to use in this paper because it evokes the primary example of discretizing open graphs (Example 13).

Definition 12 (Discrete comonad). *A comonad of an adjunction is called* ***discrete*** *if its counit is monic.*

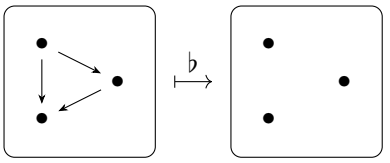
We can interpret a discrete comonad as returning largest interface $\flat x$ supported by a network x. Here is an example illustrating how the discrete graph adjunction gives rise to $\flat$.

Example 13. Consider the geometric morphism

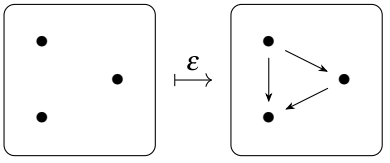
$$\mathsf{Graph} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{Set}$$

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

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668



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



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

Definition 14 (Discrete grammar). *Let* $\flat \colon \mathsf{X} \to \mathsf{X}$ *be a discrete comonad with counit* ε. *Given a grammar* (X, P), *define* $P_\flat$ *as the set containing*

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

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

4.3 Expressiveness of grammars

The following lemma uses pushout complements to characterize when (X, P) and $(\mathsf{X}, P_\flat)$ generate the same language. We then use this characterization to show that this holds for both directed and undirected hypergraphs, Petri nets, and marked Petri nets. It also holds for certain slice categories, which allows us to extend this result to typed graphs and hypergraphs.

Lemma 15. *Fix an adjunction*

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

between adhesive categories with a monic counit and L *preserving pullbacks. Take a grammar* (X, P) *and its underlying discrete grammar* $(\mathsf{X}, P_\flat)$. *If for every derived rewrite rule*

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

there is a pushout complement d *to the composite* $\flat k \rightarrow k \rightarrow k'$, *then* (X, P) *and* $(\mathsf{X}, P_\flat)$ *generate the same language.*

Proof. Suppose that $\ell' \rightsquigarrow r'$ in the rewrite relation for (X, P). That means there is a DPO diagram

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

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

in X so that the top span is in P. Then, $\ell' \rightsquigarrow r'$ in $(\mathsf{X}, P_\flat)$ as witnessed by the DPO diagram

$$\begin{array}{ccccccccc} \ell & \longleftarrow & k & \longleftarrow & \flat k & \longrightarrow & k & \longrightarrow & r \\ \downarrow & \urcorner & \downarrow & \urcorner & \downarrow & \ulcorner & \downarrow & \ulcorner & \downarrow \\ \ell' & \longleftarrow & k' & \longleftarrow & d & \longrightarrow & k' & \longrightarrow & r' \end{array}$$

Conversely, suppose that $\ell' \rightsquigarrow r'$ is generated by a rule $\ell \leftarrow \flat k \rightarrow r$ in $P_\flat$. That means there is a diagram

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

By the construction of $P_\flat$, the maps $\flat k \rightarrow \ell$ and $\flat k \rightarrow r$ both factor through the counit $\flat k \rightarrow k$ and, consequently, we get

$$\begin{array}{ccccccccc} \ell & \longleftarrow & k & \longleftarrow & \flat k & \longrightarrow & k & \longrightarrow & r \\ \downarrow & \urcorner & & & \downarrow & & & \ulcorner & \downarrow \\ \ell' & & \longleftarrow & & d & & \longrightarrow & & r' \end{array}$$

Both maps $d \rightarrow \ell'$ and $d \rightarrow r'$ factor through $k +_{\flat k} d$ which results in the diagram

$$\begin{array}{ccccc} \ell & \longleftarrow & k & \longrightarrow & r \\ \downarrow & \urcorner & \downarrow & \ulcorner & \downarrow \\ \ell' & \longleftarrow & k +_{\flat k} d & \longrightarrow & r' \end{array}$$

whose pushouts result from the pushout pasting law. This witnesses $\ell' \rightsquigarrow r'$ derived from P. □

Showing that a grammar and its underlying discrete grammar generate the same language was initially shown for directed graphs.

Example 16 Ehrig et al. (1973, Proposition 3.3). Returning to the discrete graph geometric morphism

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

any grammar (Graph, P) generates the same language as the discrete grammar $(\mathsf{Graph}, P_\flat)$.

4.3.1 Hypergraphs

We can now show for the first time that this result also holds for hypergraphs, both directed and undirected. Following (Bonchi et al. 2017), we realize hypergraphs as a presheaf topos in the following way.

Definition 17. *Define the category of directed hypergraphs* $\mathsf{DHGraph}$ *as the presheaf category* $[\mathsf{DH}^{\mathrm{op}}, \mathsf{Set}]$ *where* DH *has objects* $\mathbb{N} \times \mathbb{N} + \{N\}$ *and arrows*

$$\mathsf{DH}(N, (n, m)) := \{s_1, \ldots, s_n\} + \{t_1, \ldots, t_m\}.$$

All other homsets are trivial. For such a presheaf, the image of $(n, m) \in \mathbb{N} \times \mathbb{N}$ *gives the set of edges with in-degree n and out-degree m, the image of N gives the set of nodes, the image of each* source *map* $s_j^{\mathrm{op}} : (n, m) \rightarrow N$ *records the j-th incidence for the source of each edge, and the image of each* target *map* $t_j^{\mathrm{op}} : (n, m) \rightarrow N$ *records the j-th incidence for the target of each edge.*

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

Define the category of undirected hypergraphs HGraph *as the category of presheaves* $[\mathsf{H}^{\mathrm{op}}, \mathsf{Set}]$ *on the category* H *with objects* $\mathbb{N} + \{N\}$ *and arrows*

$$\mathsf{H}(N, n) := \{x_1, \ldots, x_n\}.$$

All other homsets are trivial. For such a presheaf, the image of $n \in \mathbb{N}$ *gives the set of edges with degree* n*, the image of* N *gives the set of nodes, and the image of each map* $x_j^{\mathrm{op}} : n \to N$ *records the* j*-th incidence for each edge.*

Both undirected hypergraphs can be *discretized* via geometric morphisms

$$\mathsf{DHGraph} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{Set} \qquad \mathsf{HGraph} \underset{R}{\overset{L}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{Set}$$

where R returns the node set and L returns a discrete hypergraph on a given set. It is straightforward to check that, in each case, the counit of the induced comonad is indeed monic. Therefore, for a grammar of directed hypergraphs $(\mathsf{DHGraph}, P)$, we have the underlying discrete grammar $(\mathsf{DHGraph}, P_\flat)$. This holds too for undirected hypergraphs.

Proposition 18. *A grammar of directed hypergraphs* $(\mathsf{DHGraph}, P)$ *and its underlying discrete grammar* $(\mathsf{DHGraph}, P_\flat)$ *have the same language.*

Proof. Consider a derived rewrite rule

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

By Lemma 15, it suffices to find a hypergraph d that is a pushout complement to $\flat k \to k \to k'$. Define d by $d(N) := k'(N)$ and $d(n, m) := k'(n, m) - k(n, m)$ for each in-degree, out-degree pair (n, m). Recalling that the directed hypergraph $\flat k$ has no edges and the same nodes as k, we have pushouts

$$\begin{array}{ccc} k(n,m) & \longleftarrow & \flat k(n,m) \\ \downarrow & \ulcorner & \downarrow \\ k'(n,m) & \longleftarrow & d(n,m) \end{array} \qquad \begin{array}{ccc} k(N) & \longleftarrow & \flat k(N) \\ \downarrow & \ulcorner & \downarrow \\ k'(N) & \longleftarrow & d(N) \end{array}$$

ensuring that d is the pushout complement we sought. □

Proposition 19. *A grammar of undirected hypergraphs* (HGraph, P) *and its underlying discrete grammar* $(\mathsf{HGraph}, P_\flat)$ *have the same language.*

Proof. The proof follows the same reasoning as the directed hypergraph case. □

4.3.2 Petri nets, unmarked, and marked

Petri nets have various definitions in the literature, particularly their morphisms. Some authors define morphisms to capture the *behavior* of Petri nets (Baez and Master 2020). Our interests lie in the structure of Petri nets as opposed to the behavior, so we take the following definition as used by Johnstone et al. (2007).

Definition 20 (Petri nets). *A* ***Petri net*** *is a tuple* $N := (P, T, s, t)$ *where* P *is a set of places,* T *is a set of* transitions, *and* $s, t : T \to P^{\oplus}$ *are maps that encode the* source *and* target *of the transitions by*

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

assigning each transition to a collection places. A ***morphism of Petri nets*** *is a pair* $(f_P, f_T)\colon N \to N'$ *where* $f_P\colon P \to P'$ *and* $f_T\colon T \to T'$ *preserve sources and targets:* $s'f_T = f_P^{\oplus}s$ *and* $t'f_T = f_P^{\oplus}t$. *The category of Petri nets and their morphisms is denoted by* PN. *This category is equivalent to the category of directed hypergraphs.*

Proposition 21. *A grammar of Petri nets* (PN, P) *and its underlying discrete grammar* $(\mathsf{PN}, P_\flat)$ *have the same language.*

Proof. The result follows from a similar argument as in Proposition 18. □

Definition 22 (Marked Petri nets). *A* ***marked Petri net*** *is a tuple* $M := (N, K, k)$ *where* $N := (P, T, s, t)$ *is a Petri net,* K *is a set of* tokens, *and* $k\colon K \to P$ *is a* marking *of the places by the tokens. A* ***morphism of marked Petri nets*** $(f_N, f_K)\colon M \to M'$ *is a Petri net morphism* $f_N\colon N \to N'$ *together with a map between token sets* $f_K\colon K \to K'$ *such that* $k'f_K = f_P k$. *The category of marked Petri nets and their morphisms is denoted by* MPN. *This is equivalent to a presheaf category* $[\mathsf{M}^{\mathrm{op}}, \mathsf{Set}]$ *for* M *the category with objects* $\mathbb{N} \times \mathbb{N} + \{P, K\}$ *and arrows* $\mathsf{M}(P, (n, m)) = \{s_1, \ldots, s_n\} + \{t_1, \ldots, t_m\}$ *plus* $\mathsf{M}(P, K) = \{k\}$.

A *discrete marked Petri net* is one without transitions or tokens. Similar to graphs and hypergraphs, there is a discrete marked Petri net adjunction.

Proposition 23. *A grammar of marked Petri nets* (MPN, P) *and its underlying discrete grammar* $(\mathsf{MPN}, P_\flat)$ *have the same language.*

Proof. The result follows from a similar argument as in Proposition 18. □

4.3.3 Typed objects

Occasionally, those working with rewriting *typify* their objects. For instance, instead of working with graphs, one might work with graphs whose edges and notes each come in multiple colors. This can be accomplished by working in an appropriate slice category. For example, suppose we wanted to work with graphs whose edges come in red r and green g colors. Add this information to a graph using a map to



so that the images of the edges and nodes of g encode the typing information. In short, we may want to be rewriting in a slice category X/x using an adhesive category X and object x.

This next result provides conditions for a grammar and its underlying discrete grammar in X/x to generate the same language. The first condition is that the "untyped grammars" (X, P) and $(\mathsf{X}, P_\flat)$ generate the same rewrite relation. To move to a grammar in X/x, there would be user-driven decisions to make, namely how the rewrite rules in P will be mapped to x.

Proposition 24. *Fix an adjunction* $L \dashv R\colon \mathsf{X} \to \mathsf{A}$ *between adhesive categories with a monic counit and* L *preserving pullbacks. Fix an object* x *in* X. *Consider both the comonad* $\flat := LR$ *and the induced adjunction between slice categories* $L_x \dashv R_x\colon \mathsf{X}/x \to \mathsf{A}/Rx$ *that also has a monic counit and* L_x *preserving pullbacks. Let* $(\mathsf{X}/x, P_x)$ *be any grammar,* $(\mathsf{X}/x, P_{x,\flat})$ *be the underlying discrete grammar, and* (X, P) *and* $(\mathsf{X}, P_\flat)$ *be their projections into* X.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

If for every derived rewrite rule

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

in (X, P) *there is a pushout complement* d *to the composite* $\flat k \to k \to k'$*, then* $(\mathsf{X}/x, P_x)$ *and* $(\mathsf{X}/x, P_{x,\flat})$ *generate the same rewrite relation.*

Proof. Recalling that pushouts in X/x are computed by projecting to X, this result follows from a similar argument as in Lemma 15. □

Corollary 25. *Grammars and their underlying discrete grammars generate the same rewrite relation for typed directed graphs, typed directed hypergraphs, and typed undirected hypergraphs.*

4.4 Inductive rewriting systems

In the networks perspective, a grammar (X, P) can be thought of as comprising an adhesive category X of closed networks and a set of rules P stating how to revise the closed networks. For example, one might revise an network of resistors by replacing a series of resistors with a single resistor. An important part of our construction involves decomposing a closed network x into other networks $x_1, \ldots, x_n$ that are somehow connected. Structured cospans provide a way to form these connections, hence we want X to fit into an appropriate adjunction $L \dashv R$. Because our construction uses structured cospans, we need a way to represent a closed network with structured cospans. We turn a closed network x into the structured cospan $L0 \to x \leftarrow L0$ with an empty interface. Note that here 0 denotes an initial object in A, and $L0$ is an initial object in X because L is a left adjoint.

The particular decomposition we use is determined by the grammar. That is, we start with a grammar (X, P) where X can fit into an adjunction $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ with a monic counit satisfying the adhesivity condition. This gives a discrete comonad $\flat := LR$, which allows us to form the discrete grammar $(\mathsf{X}, P_\flat)$ as in Definition 14.

Finally, we need to define a particular grammar used to generate our inductive viewpoint of rewriting.

Definition 26. *Fix an adjunction* $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ *and a grammar* $G := (\mathsf{X}, P)$*. Define* $G' = ({}_L\mathsf{StrCsp}, P'_\flat)$ *to be a structured cospan grammar where* $P'_\flat$ *contains the rule*

$$\begin{array}{ccccc} & & \ell & & \\ & \nearrow & \uparrow & \nwarrow & \\ L0 & \longrightarrow & \flat k & \longleftarrow & \flat k \\ & \searrow & \downarrow & \swarrow & \\ & & r & & \end{array}$$

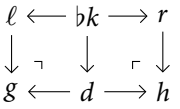
for each rule $\ell \leftarrow \flat k \to r$ *of* $P_\flat$*.*

Applying the language construction $\mathbf{Lang}(G')$, then, encodes $\mathrm{Lang}(G)$ in its 2-cells, which are generated by the rules in $P_\flat$. Since $\mathrm{Lang}(G)$ is generated by the rewrite relation for G, this forms the inductive viewpoint of the rewrite relation.

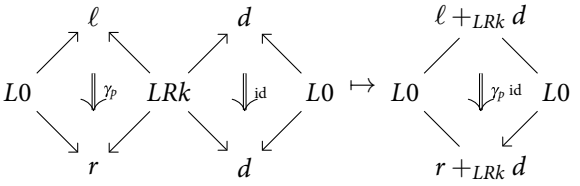
Theorem 27. *Let* $L \dashv R \colon \mathsf{X} \to \mathsf{A}$ *be an adjunction that satisfies the adhesivity condition and induces a discrete comonad* $\flat$*. Given any grammar* $G := (\mathsf{X}, P)$ *whose language is the same as its underlying discrete grammar* $G_\flat := (\mathsf{X}, P_\flat)$*. There is an equivalence between* $\mathrm{Lang}(G)$ *and* $\mathbf{Lang}(G')(L0, L0)$*.*

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

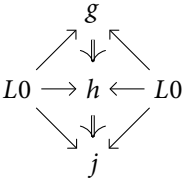
Proof. We show sufficiency by inducting on the length of the generating arrows of Lang(G), which come from the rewrite relation $\rightsquigarrow^*$. If $g \rightsquigarrow^* h$ in a single step, meaning that there is a diagram



in X whose top row is a rewrite rule in P, then the desired 2-cell is the horizontal composition of



where $g \cong l +_{\flat k} d$ and $h \cong r +_{\flat k} d$. Then, the 2-cell for a derivation $g \rightsquigarrow^* h \rightsquigarrow j$ is the vertical composition of

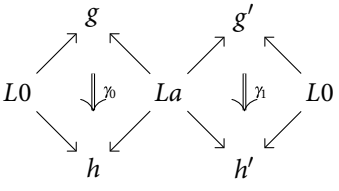


The top square is from $g \rightsquigarrow^* h$ and the second from $h \rightsquigarrow j$.

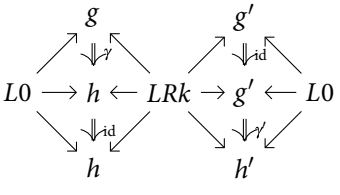
Conversely, we proceed by structural induction to show that given an arrow in the hom-category $\mathbf{Lang}(G')(L0, L0)$, there is a corresponding derivation per the rewrite relation. The base case is the generating 2-cells, which holds by construction. Next, we show that both horizontal and vertical composition preserve $\rightsquigarrow^*$. Vertical composition preserves $\rightsquigarrow^*$ due to the transitivity of the rewrite relation. Horizontal composition must have the form

$$\mathbf{Lang}(G')(L0, La) \times \mathbf{Lang}(G')(La, L0) \to \mathbf{Lang}(G')(L0, L0).$$

So given



where $g \rightsquigarrow^* h$ and $g' \rightsquigarrow^* h'$ then we get $g +_a g' \rightsquigarrow^* h +_a h'$ by realizing the horizontal composition as



Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

composing horizontally, then performing this rewrite by first applying $g \rightsquigarrow^* h$ to g to get $g +_{La} g' \rightsquigarrow^* h +_{La} g'$ and then applying $g' \rightsquigarrow^* h'$ to g' to get $g +_{La} g' \rightsquigarrow^* h +_{La} h'$. □

As a corollary, we have that Theorem 27 applies to grammars in the categories of directed hypergraphs, undirected hypergraphs, unmarked Petri nets, marked Petri nets, and their typed variants. It also applies to typed graphs so applies to the ZX-calculus (Cicala 2018a).

5. Conclusion

We have introduced a new category of structured cospans ${}_L\mathsf{StrCsp}$ and characterized when it is adhesive and when it is a topos. Under these conditions, ${}_L\mathsf{StrCsp}$ admits a theory of rewriting. Next, we have provided a condition for a grammar and its underlying discrete grammar to induce the same language. Using that condition, we introduced an inductive viewpoint for rewriting, thus allowing for proof strategies involving structural induction. In particular, we have shown this holds for directed hypergraphs, undirected hypergraphs, their typed variants, for Petri nets, marked Petri nets, and the ZX-calculus.

Competing interests. The author declares none.

References

Baader, F. and Nipkow, T. (1998). *Term Rewriting and All that*, Cambridge University Press.

Baez, J. and Courser, K. (2020). Structured cospans. *Theory and Applications of Categories* 35 1771–1822.

Baez, J., Courser, K. and Vasilakopoulou, C. (2022). Structured versus decorated cospans. *Compositionality* **4** 3.

Baez, J. and Fong, B. (2018). A compositional framework for passive linear networks. *Theory and Applications of Categories* **33** 1158–1222.

Baez, J. and Master, J. (2020). Open Petri nets. *Mathematical Structures in Computer Science* **30** (3) 314–341.

Baldan, P., Corradini, A., Ehrig, H. and Heckel, R. (2001). Compositional modeling of reactive systems using open nets. In: Larsen, K. G. and Nielsen, M. (eds.) *CONCUR 2001 — Concurrency Theory*, Berlin, Heidelberg, Springer Berlin Heidelberg, 502–518.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2016). Rewriting modulo symmetric monoidal structure. In: *Proceedings of the 31st Annual ACM-IEEE Symposium on Logic in Computer Science (LICS 2016)*, vol. 10, New York, ACM.

Bonchi, F., Gadducci, F., Kissinger, A., Sobociński, P. and Zanasi, F. (2017). Confluence of graph rewriting with interfaces. In: *Programming Languages and Systems: 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings 26*, Springer, 141–169.

Cicala, D. (2018a). Categorifying the ZX-calculus. In: Coecke, B. and Kissinger, A. (eds.) *Proceedings 14th International Conference on Quantum Physics and Logic, Nijmegen, The Netherlands, 3-7 July 2017*, vol. 266 of Electronic Proceedings in Theoretical Computer Science, Open Publishing Association, 294–314.

Cicala, D. (2018b). Spans of cospans. *Theory and Applications of Categories* 33 131–147.

Cicala, D. and Courser, K. (2018). Spans of cospans in a topos. *Theory and Applications of Categories* **33** 1–22.

Courser, K. (2020). *Open systems: A double categorical perspective*. Phd thesis, Riverside, University of California.

Dixon, L. and Kissinger, A. (2013). Open-graphs and monoidal theories. Mathematical Structures in Computer Science **23** (2) 308–359.

Ehrig, H. and König, B. (2004). Deriving bisimulation congruences in the DPO approach to graph rewriting. In: Walukiewicz, I. (ed.) *Foundations of Software Science and Computation Structures*, Berlin, Heidelberg, Springer Berlin Heidelberg, 151–166.

Ehrig, H., Pfender, M. and Schneider, H. (1973). Graph-grammars: An algebraic approach. In: *14th Annual IEEE Symposium on Switching and Automata Theory (Univ. Iowa, Iowa City, Iowa, 1973)*, Northridge, CA, IEEE, 167–180.

Fong, B. (2015). Decorated cospans. Theory and Applications of Categories **30** 1096–1120.

Gadducci, F. and Heckel, R. (1998). An inductive view of graph transformation. In: *Recent Trends in Algebraic Development Techniques (Tarquinia 1997)*, vol. 1376 of Lecture Notes in Computer Science, Berlin, Springer, 223–237.

Johnstone, P. (2002). *Sketches of an Elephant: A Topos Theory Compendium*, vol. 2, Oxford University Press.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668

Johnstone, P., Lack, S. and Sobociński, P. (2007). Quasitoposes, quasiadhesive categories and Artin glueing. In *Algebra and Coalgebra in Computer Science*, vol. 4624 of Lecture Notes in Computer Science, Berlin, Springer, 312–326.

Lack, S. and Sobociński, P. (2004). Adhesive categories. In: *Foundations of Software Science and Computation Structures*, vol. 2987 of Lecture Notes in Computer Science, Berlin, Springer, 273–288.

Lack, S. and Sobociński, P. (2006). Toposes are adhesive. In: *Graph Transformations*, vol. 4178 of Lecture Notes in Computer Science, Berlin, Springer, 184–198.

Rodgers, P. and King, P. (1997). A graph-rewriting visual language for database programming. *Journal of Visual Languages & Computing* **8** (5–6) 641–674.

Sassone, V. and Sobociński, P. (2004). *Congruences for Contextual Graph-Rewriting*, Citeseer.

Sassone, V. and Sobociński, P. (2005). A congruence for Petri nets. *Electronic Notes in Theoretical Computer Science* **127** (2) 107–120.

Street, R. (1976). Limits indexed by category-valued 2-functors. *Journal of Pure and Applied Algebra* **8** (2) 149–181.

Wraith, G. (1974). Artin glueing. *Journal of Pure and Applied Algebra* **4** 345–348.

Cite this article: Cicala D (2026). Rewriting structured cospans. *Mathematical Structures in Computer Science* **36**(e28), 1–19. https://doi.org/10.1017/S0960129526100668

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 12 Nov 2026 at 08:05:19, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129526100668