Knowledge Representation in Bicategories of Relations

Evan Patterson

Department of Statistics, Stanford University

Abstract

We introduce the *relational ontology log*, or *relational olog*, a knowledge representation system based on the category of sets and relations. It is inspired by Spivak and Kent's *olog*, a recent categorical framework for knowledge representation. Relational ologs interpolate between ologs and description logic, the dominant formalism for knowledge representation today. In this paper, we investigate relational ologs both for their own sake and to gain insight into the relationship between the algebraic and logical approaches to knowledge representation. On a practical level, we show by example that relational ologs have a friendly and intuitive—yet fully precise—graphical syntax, derived from the string diagrams of monoidal categories. We explain several other useful features of relational ologs not possessed by most description logics, such as a *type system* and a rich, flexible notion of *instance data*. In a more theoretical vein, we draw on categorical logic to show how relational ologs can be translated to and from logical theories in a fragment of first-order logic. Although we make extensive use of categorical language, this paper is designed to be self-contained and has considerable expository content. The only prerequisites are knowledge of first-order logic and the rudiments of category theory.

1. Introduction

The representation of human knowledge in computable form is among the oldest and most fundamental problems of artificial intelligence. Several recent trends are stimulating continued research in the field of knowledge representation (KR). The birth of the Semantic Web [BHL01] in the early 2000s has led to new technical standards and motivated new machine learning techniques to automatically extract knowledge from unstructured text [Nic+16]. In the scientific community, successful knowledge bases like the Gene Ontology [Ash+00] have inspired a proliferation of ontologies across biology and biomedicine [Noy+09]. This development belongs to a general trend towards greater openness and interconnectivity in science. Optimists dream of a future science in which all scientific knowledge is open, online, and interpretable by machines [Nie12].

Description logic is the dominant formalism for knowledge representation today. In particular, the language OWL (Web Ontology Language), a W3C standard underlying the Semantic Web, is a description logic [Gra+08]. Description logics are logical calculi

designed specifically for knowledge representation. They lie somewhere between propositional logic and first-order predicate logic, striking a trade-off between computational tractability and expressivity.

In parallel with the invention of description logic and the Semantic Web, a mostly disjoint community of mathematicians, physicists, and computer scientists have discovered that category theory, popularly known for its abstruseness, is useful not just for describing abstract mathematical structures, but for modeling such diverse real-world phenomena as databases, programming languages, electrical circuits, and quantum mechanics [Spi12; LS88; BF15; AC04]. The ethos of this research program is that category theory can serve as a general-purpose modeling language for science and engineering. Having internalized this perspective, it is but a short step to contemplate a general-purpose knowledge representation system based on category theory. In this spirit, Spivak and Kent have recently introduced the *ontology log* (or *olog*), a simple and elegant categorical framework for knowledge representation [SK12].

An objective of this paper is to understand the relationship between the logical and algebraic approaches to knowledge representation. To that end, we introduce a third knowledge representation formalism that interpolates between description logic and ontology logs. We call it the *relational ontology log*, or *relational olog*. Spivak and Kent's ologs, which we sometimes call *functional ologs* to avoid confusion, are based on **Set**, the archetypal category of sets and functions. Relational ologs are based on **Rel**, the category of sets and relations. That may sound a small difference, since functions and relations are often interchangeable, but it leads to very different modes of expression. Functional ologs achieve their expressivity through categorical limits and colimits (products, pullbacks, pushforwards, etc.), while relational ologs rely mainly on relational algebra (intersections, unions, etc.). In this sense, relational ologs are actually closer to description logic than to functional ologs.

Practitioners of description logic will find in relational ologs several useful features not possessed by most existing KR systems, including OWL. Some of these features are awkward to handle in a purely logical system; all emerge automatically from the categorical framework. First, functors allow *instance data* to be associated with an ontology in a mathematically precise way. Instance data can be interpreted as a relational or graph database or can take more exotic forms. Second, relational ologs are by default *typed*. We argue that types, if used judiciously, can mitigate the maintainability challenges posed by the open world semantics of description logic. Finally, relational ologs have a friendly and intuitive—yet fully precise—*graphical syntax*, derived from the string diagrams of monoidal categories. We expect that this graphical language will appeal to technical and non-technical users alike.

How to read this paper   We have tried to write a paper that is accessible to a diverse audience. All Remarks and Appendices are technical and can be skipped on a first reading. The mathematical prerequisites are limited as follows. We assume the reader is familiar with the syntax and semantics of first-order logic. No prior knowledge of description logic is required. We expect the reader to know the "big three" concepts of category theory—category, functor, and natural transformation—but we do not assume knowledge of categorical logic or monoidal categories and their graphical languages. References for further reading are provided where appropriate.

Readers who prefer to begin with an extended example may proceed immediately to Section 5.1, referring to Section 3 as needed to understand the graphical notation. The core of the paper, explaining the categorical-relational approach to knowledge representation, is Sections 3 to 7. The other sections develop extensions of our methodology and make connections to other branches of mathematics and knowledge representation.

Organization of paper   In the next section, we review description logic as a computationally tractable subset of first-order logic and describe several widely used description logics. In Section 3, we introduce **Rel**, the category of sets and relations, and use it to illustrate the general concepts of monoidal categories and their graphical languages. We also make initial contact with the basic notions of description logic. Motivated by **Rel**, in Section 4 we present the *bicategory of relations*, a categorical abstraction of relational algebra invented by Carboni and Walters. Section 5 defines a *relational olog* to be a finitely presented bicategory of relations and illustrates with an extended example. Sections 6 and 7 discuss the implications of instance data and types for knowledge representation. In Section 8, we take a sojourn into *categorical logic*, proving that regular logic is the internal language of bicategories of relations. This result establishes a formal connection between relational ologs and a fragment of typed first-order logic. In Section 9, we introduce the *distributive relational olog*, an extension of the relational olog with high expressivity. In the final Section 10, we comment on the philosophy of categorical knowledge representation and suggest directions for future research. The two Appendices bring mathematical rigor to the informal discussion of categorical logic in the main text.

2. Description logic

Early knowledge representation systems, based on semantic networks or frames, often lacked a formal semantics. The intended meanings of the elements of such systems were defined only implicitly or operationally by the inference algorithms that manipulated them. As a result, researchers found it difficult to reason generally about these systems, independent of any specific implementation. Arguments were advanced that knowledge representation should be grounded in formal logic [Woo75]. First-order logic, ever the "default" logical system, seems like a natural place to start.

Description logic (DL) is motivated by the deficiencies of first-order logic as a foundation for knowledge representation. Chief among these is computational intractability: first-order logic, while quite expressive, is undecidable. The basic description logics are subsets of first-order logic designed to be decidable (although not always in polynomial time). The tradeoff between expressivity and tractability was emphasized by the earliest papers on description logic [BL84]. Another point, less frequently mentioned, is that description logic is simpler and more user friendly than first-order logic. As we will see, its syntax suppresses variables, both bound and free, and imposes strict limits on the logical sentences that can be formed. Given that most users of knowledge representation systems are domain experts in scientific or business fields, not professional mathematicians, it is important that KR formalisms be easily interpretable and maintainable. A knowledge base consisting of a collection of arbitrary first-order sentences will probably not meet this requirement.

2.1. Review of description logic

In this section, we briskly review description logic. General introductions to description logic include the survey [KSH12] and the textbook chapter [BL04, Ch. 9] by Brachman and Levesque. A comprehensive reference is the *Description Logic Handbook* [Baa+07]. For the perspectives of the bioinformatics and Semantic Web communities, see [RB11] and [HKR09], respectively.

Description logic uses a special nomenclature to specify the features possessed by a given system. The base system, from which most others are derived, is called $\mathcal{AL}$ (for Attributive Concept Language). Given a collection of *atomic concepts*, denoted A, and *atomic roles*, denoted R or S, the *concept* descriptions of $\mathcal{AL}$ are well-formed terms of the grammar:

$$
\begin{array}{llll}
C, D & ::= & A \mid & \text{(atomic concept)} \\
& & \top \mid & \text{(universal concept)} \\
& & \bot \mid & \text{(bottom concept)} \\
& & \neg A & \text{(atomic negation)} \\
& & C \sqcap D \mid & \text{(intersection)} \\
& & \forall R.C \mid & \text{(value restriction)} \\
& & \exists R.\top & \text{(limited existential quantification).}
\end{array}
$$

Note that negating arbitrary concepts is not allowed in $\mathcal{AL}$. Concepts and roles are interpreted as unary and binary predicates in first-order logic:

$$
\begin{array}{lll}
(\neg A)(x) & \text{iff} & \neg A(x) \\
(C \sqcap D)(x) & \text{iff} & C(x) \wedge D(x) \\
(\forall R.C)(x) & \text{iff} & \forall y.(R(x,y) \to C(y)) \\
(\exists R.\top)(x) & \text{iff} & \exists y.R(x,y)
\end{array}
$$

A *terminological box* or *TBox* is a collection of *terminological axioms* of form

$$
C \sqsubseteq D \qquad \text{or} \qquad C \equiv D,
$$

interpreted as the first-order sentences

$$
\forall x.(C(x) \to D(x)) \qquad \text{or} \qquad \forall x.(C(x) \leftrightarrow D(y)).
$$

An *assertional box* or *ABox* is a collection of *assertional axioms* of form

$$
C(a) \qquad \text{or} \qquad R(a,b),
$$

where a, b are names of individuals. A *knowledge base* or *ontology* in description logic consists of a TBox and an ABox. Given the above translations into first-order logic, there is an obvious notion of an *interpretation* or *model* of a knowledge base. Thus description logic inherits a model-theoretic semantics from first-order logic.

More expressive description logics are obtained by adjoining to $\mathcal{AL}$ additional concept and role constructors, identified by script letters like $\mathcal{C}$ and $\mathcal{U}$. The literature describes countless such extensions; Table 1 lists the most important ones. As a warning, a few identifiers (like $\mathcal{F}$ and $\mathcal{R}$) are not used consistently across the literature.

ID	Name	DL Syntax	FOL Interpretation
$\mathcal{C}$	Concept negation	$\neg C$	$\neg C(x)$
$\mathcal{U}$	Concept union	$C \sqcup D$	$C(x) \vee D(x)$
$\mathcal{E}$	Full existential quantification	$\exists R.C$	$\exists y.(R(x,y) \wedge C(y))$
$\mathcal{H}$	Role axioms	$R \sqsubseteq S$	$\forall x,y.(R(x,y) \rightarrow S(x,y))$
		$R \equiv S$	$\forall x,y.(R(x,y) \leftrightarrow S(x,y))$
$\mathcal{O}$	Nominals (concept literals)	$\{a_1,\ldots,a_n\}$	$\{a_1,\ldots,a_n\}$
$\mathcal{I}$	Inverse roles	R^-	$R^-(x,y) \leftrightarrow R(y,x)$
$\mathcal{F}$	Functional role	$\leq 1R$	see below
$\mathcal{N}$	Number restriction	$\geq nR$	see below
		$= nR$	
		$\leq nR$	
$\mathcal{Q}$	Qualified number restriction	$\geq nR.C$	see below
		$= nR.C$	
		$\leq nR.C$	
N/A	Role intersection	$R \sqcap S$	$R(x,y) \wedge S(x,y)$
N/A	Role union	$R \sqcup S$	$R(x,y) \vee S(x,y)$
N/A	Role composition	$R \circ S$	$\exists y.R(x,y) \wedge S(y,z)$
$\mathcal{R}$	Regular role inclusion	$R_1 \circ \cdots \circ R_n \sqsubseteq S$	see below
$(\mathcal{D})$	Concrete domains (data types)	varies	varies

Table 1: Summary of $\mathcal{AL}$ language extensions

Several DL constructs in Table 1 deserve elaboration. The qualified number restriction ($\mathcal{Q}$) term $\geq nR.C$ (respectively $\leq nR.C$) denotes the class of elements related by R to at least n (respectively at most n) elements of class C. In first-order logic,

$$(\geq nR.C)(x) \quad \text{iff} \quad \exists y_1,\ldots,y_n.\left(\bigwedge_{1 \leq i \leq n} (R(x,y_i) \wedge C(y_i)) \wedge \bigwedge_{1 \leq i < j \leq n} y_i \neq y'_j \right)$$

$$(\leq nR.C)(x) \quad \text{iff} \quad \forall y_1,\ldots,y_{n+1}.\left(\bigwedge_{1 \leq i \leq n} (R(x,y_i) \wedge C(y_i)) \rightarrow \bigvee_{1 \leq i < j \leq n+1} y_i = y'_j \right).$$

Number restriction ($\mathcal{N}$) and functional roles ($\mathcal{F}$) are special cases of qualified number restriction. Concrete domains ($(\mathcal{D})$) refer to data types, such as natural numbers or real numbers, and operations on them, such as addition and multiplication. We return to the topic of data types in Section 7.

Most descriptions logics do not allow arbitrary intersection, union, or composition of roles. However, composition-based regular role inclusion ($\mathcal{R}$) is widely used. System $\mathcal{R}$ allows axioms of form $R_1 \circ \cdots \circ R_n \sqsubseteq S$, where $R_1,\ldots,R_n$ are atomic roles, provided there are no cycles between axioms. This acyclicity requirement, which we will not make precise, leads to favorable computational properties. Note that $\mathcal{R}$ is sometimes taken to include additional, ad hoc features like reflexivity, "local" reflexivity, irreflexivity, and disjoint roles [HKS05; HKS06].

A few description logics are privileged in theory or practice. The minimal language $\mathcal{AL}$ is too inexpressive for most applications. The central language in the theory of description

logic is $\mathcal{ALC}$. It is logically equivalent to $\mathcal{ALUE}$, although the shorter name $\mathcal{ALC}$ is preferred. In a break with the standard nomenclature, the language $\mathcal{S}$ is $\mathcal{ALC}$ plus transitive roles. The Web Ontology Languages are derived from system $\mathcal{S}$. For example, OWL 1 Lite corresponds to $\mathcal{SHIF}(\mathcal{D})$, OWL 1 DL to $\mathcal{SHION}(\mathcal{D})$, and OWL 2 DL to $\mathcal{SROIQ}(\mathcal{D})$.

2.2. Structure of description logic

To put the subsequent developments in context, we make a few observations about the structure of description logic. Since description logic is not a single logical system, but rather a large federation of systems, it is difficult to make broad generalizations. Nevertheless, some general themes can be discerned.

An obvious syntactic difference between description logic and first-order logic is that the former is *point-free* while the latter is not. By "point-free" we mean that the concept and role constructors of description logic suppress all variables, free and bound. First-order logics without variables *do* exist—Tarksi, for example, studied such systems in his last major work [TG87]—but, outside of description logic, they are rare in research and in practice. In this respect, relational ologs are like description logic: both the textual and graphical syntaxes of relational ologs are point-free.

Description logics characteristically impose strict limitations on how concepts and roles may be combined. Thus, not all first-order sentences are expressible in description logic. The same is true of relational ologs: we shall see that when relational ologs are interpreted as first-order theories (Section 8), not all first-order sentences are expressible.

Moreover, there are structural similarities between the first-order sentences that *are* expressible in the two formalisms. In description logic, terminological axioms $C \sqsubseteq D$ and $R \sqsubseteq S$ typically translate into first-order sentences of form

$$\forall x_1 \cdots \forall x_n (\varphi \to \psi),$$

where φ and ψ are formulas containing only the connectives and quantifiers $\wedge, \vee, \top, \bot, \exists$. (Depending on the language, exceptions can arise from value restrictions $\forall R.C$ and number restrictions like $\leq nR.C$. However, these constructors are acceptable in axioms of form $\forall R.C \sqsubseteq \top$ or $\forall R.C \sqsubseteq \bot$. Concept negations $\neg C$ also present exceptions.) The logical system just described is called *coherent logic*. The weaker system of *regular logic* is obtained when φ and ψ are further restricted to the connectives and quantifiers $\wedge, \top, \exists$. We shall see that regular logic and coherent logic are closely connected to relational ologs (Sections 8 and 9).

3. The category of relations

In this section we introduce **Rel**, the category of sets and relations. Although the reader is doubtless familiar with sets and relations, we think it helpful to start the development in this very concrete setting. We will introduce monoidal categories and their graphical languages by equipping **Rel** with various categorical structures, such as a monoidal product, diagonals and codiagonals, and a dagger operator. These structures on **Rel**

motivate the more abstract "categories of relations" needed for knowledge representation (Section 4). We will also make initial contact with description logic.

Our presentation draws on the physics-oriented survey by Coecke and Paquette [CP10], where **Rel** is viewed as a "quantum-like" category, in contrast to the "classical-like" category **Set**. The excellent surveys [BS10] and [Sel10] also provide more detail about monoidal categories and their applications and graphical languages. General introductions to category theory, in order of increasing sophistication, are [LS09; Spi14; Awo10; Lei14; Rie16; Mac98].

Definition. The *category of sets and relations*, denoted by **Rel**, is the category whose objects are sets and whose morphisms $R : X \to Y$ are subsets $R \subseteq X \times Y$. The *composition* of $R : X \to Y$ and $S : Y \to Z$, written $R \cdot S : X \to Z$ or $RS : X \to Z$, is given by

$$xRSz \qquad \text{iff} \qquad \exists y \in Y : xRy \wedge yRz,$$

where xRy means that $(x, y) \in R$. For any set X, the *identity* morphism 1_X is the diagonal relation:

$$x(1_X)x' \qquad \text{iff} \qquad x = x'.$$

The notion of composition of relations is natural and important. Notice that when R and S are (graphs of) functions, RS is the usual composition of functions. Also, the identity morphism is the usual identity function. As a result, **Set**, the category of sets and functions, is a subcategory of **Rel**.

Remark. As illustrated by the definition, we compose morphisms in left-to-right or "diagrammatic" order. We make this choice for consistency with the graphical syntax, which is read from left to right. It is also consistent with the notation xRy for $(x, y) \in R$.

Unlike **Set**, the category **Rel** is a special kind of a 2-category.

Definition. A category $\mathcal{C}$ is a *locally posetal 2-category* if between any two morphisms $f, g : A \to B$ with common domain and codomain, there exists at most one *2-morphism*, written

$$A \underset{g}{\overset{f}{\Rightarrow}} B$$

or more succinctly $f \Rightarrow g$, together with operations of *vertical composition*,

$$A \overset{f}{\underset{h}{\Rrightarrow}} B \qquad \rightsquigarrow \qquad A \underset{h}{\overset{f}{\Rightarrow}} B \,,$$

and *horizontal composition*,

$$A \underset{g}{\overset{f}{\Rightarrow}} B \underset{k}{\overset{h}{\Rightarrow}} C \qquad \rightsquigarrow \qquad A \underset{g \cdot k}{\overset{f \cdot h}{\Rightarrow}} C \,,$$

such that each hom-set $\mathcal{C}(A, B)$ is a poset (partially ordered set) under the relation $\Rightarrow$.

In **Rel**, we stipulate that $R \Rightarrow S$ if there is a set containment $R \subseteq S$. Vertical composition simply says that set containment is transitive. More interestingly, horizontal composition says that containment is preserved by composition of relations: if $R \subseteq S$ and $T \subseteq U$, then $R \cdot T \subseteq S \cdot U$. Furthermore, the hom-sets are posets because $R \subseteq R$ and also $R = S$ whenever $R \subseteq S$ and $S \subseteq R$. Thus **Rel** is a locally posetal 2-category. In this context, the symbol $\Rightarrow$ has a happy double meaning: we can read $\Rightarrow$ as a generic 2-morphism or as *logical implication*. In the terminology of description logic, 2-morphisms are subsumptions.

3.1. Monoidal category

We will make **Rel** into a monoidal category by equipping it with the Cartesian product. We first state the general definition of a monoidal category.

Definition. A (strict) *monoidal category* $(\mathcal{C}, \otimes, I)$ is a category $\mathcal{C}$ together with a functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, called the *monoidal product*, and an object I, called the *monoidal unit*, such that $(\otimes, I)$ behaves like a monoid on the objects and morphisms of $\mathcal{C}$, in the following sense. For objects A, B, C, we have

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C, \qquad A \otimes I = I \otimes A = I,$$

and for morphisms f, g, h, we have

$$f \otimes (g \otimes h) = (f \otimes g) \otimes h, \qquad f \otimes 1_I = 1_I \otimes f = f.$$

More explicitly, functorality of the monoidal product $\otimes$ means that for any objects A, B,

$$1_A \otimes 1_B = 1_{A \otimes B}$$

and for any morphisms $f : A \to B$, $g : B \to C$, $h : D \to E$, $k : E \to F$,

$$(f \cdot g) \otimes (h \cdot k) = (f \otimes h) \cdot (g \otimes k).$$

Let us immediately introduce the graphical language of *string diagrams* that is associated with any monoidal category. In this language, objects are represented by wires and morphisms are represented by boxes with incoming and outgoing wires. A generic morphism $f : A \to B$ is represented as

.
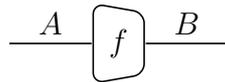
The composite $fg : A \to C$ of $f : A \to B$ and $g : B \to C$ is
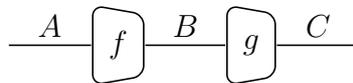
and the monoidal product $f \otimes g : A \otimes B \to C \otimes D$ of $f : A \to B$ and $g : C \to D$ is
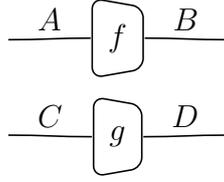
Identity morphisms are represented specially as a bare wire:

$$1_A \quad = \quad \underline{A}.$$

Each time we introduce a new monoidal structure, we will augment the graphical language accordingly.

String diagrams are among the most beautiful aspects of the theory of monoidal categories. Unlike the diagrams and flowcharts found throughout the engineering literature, which have no formal meaning, string diagrams provide a formal calculus for reasoning in monoidal categories. More precisely, *coherence theorems* guarantee that string diagrams constitute a *sound* and *complete* calculus for equational reasoning with morphisms in a monoidal category. Coherence theorems are emphasized by Selinger's comprehensive survey [Sel10]. All string diagrams in this paper are drawn by the author's (highly experimental) library for computational category theory [Pat17].

The Cartesian product makes **Rel** into a monoidal category in the most straightforward way. (Later we will see that it is not the only interesting monoidal product on **Rel**.) On objects, define

$$X \otimes Y := X \times Y = \{(x, y) : x \in X, y \in Y\}$$

and given morphisms $R : X \to Y$ and $S : Z \to W$, define $R \otimes S : X \otimes Z \to Y \otimes W$ by

$$(x, z)(R \otimes S)(y, w) \qquad \text{iff} \qquad xRy \wedge zSw.$$

The monoidal unit is any singleton set, which we write as $I = \{*\}$.

Remark. Technically, **Rel** is not a *strict* monoidal category, as defined above, because the Cartesian product is not strictly associative: $X \times (Y \times Z) \neq (X \times Y) \times Z$. Of course, there is a natural isomorphism $X \times (Y \times Z) \cong (X \times Y) \times Z$, mapping $(x, (y, z))$ to $((x, y), z)$, that allows us to identify these two sets. Similarly, $X \times I \neq X$ but there is a natural isomorphism $X \times I \cong X$ that identifies $(x, *)$ with x. Such considerations lead to the general definition of a monoidal category, where strict associativity and units are replaced with *associator* and *unitor* natural isomorphisms, subject to some coherence conditions. However, the coherence theorem for monoidal categories [Mac63] ensures that a general monoidal category is monoidally equivalent to some strict monoidal category, called its *strictification*. As a result, we adopt the common practice of suppressing associators and unitors, effectively replacing every monoidal category by its strictification. Note that the graphical language implicitly performs this strictification. Incidentally, the abstract categories of relations we construct in Section 8 will actually be strict monoidal categories.

As in most monoidal categories in encountered in practice, in **Rel** the order of inputs and outputs can be freely exchanged.

Definition. A monoidal category $(\mathcal{C}, \otimes, I)$ is a *symmetric monoidal category* if there is a natural family of isomorphisms

$$\sigma_{A,B} : A \otimes B \to B \otimes A, \qquad A, B \in \mathcal{C},$$

called *braidings*, satisfying $\sigma_{A,B}^{-1} = \sigma_{B,A}$.

In the graphical language, braidings are represented by crossed wires:

The braidings in **Rel** are defined by

$$(x, y) \, \sigma_{X,Y} \, (y', x') \qquad \text{iff} \qquad x = x' \wedge y = y'.$$

In **Rel**, unlike in **Set**, there is a fundamental duality between inputs and outputs under which any input can be turned into an output and vice versa. This duality is captured abstractly by the following definition.

Definition. A symmetric monoidal category $(\mathcal{C}, \otimes, I)$ is a *compact closed category* if for every object $A \in \mathcal{C}$, there is an object A^*, the *dual* of A, and a pair of morphisms $\eta_A : I \to A^* \otimes A$ and $\epsilon_A : A \otimes A^* \to I$, the *unit* and *counit* respectively, which satisfy the *triangle* or *zig-zag* identities:

The prototypical example of a compact closed category is $(\mathbf{Vect}_k, \otimes)$, the category of finite-dimensional vector spaces (over a fixed field k) and linear maps, equipped with the tensor product. The monoidal unit is $I = \mathbb{C}$, the one-dimensional vector space. As expected, the dual A^* of a vector space A is the space of linear maps $A \to \mathbb{C}$. The unit $\eta_A : I \to A^* \otimes A$ maps c to $c 1_A$ and the counit $\epsilon_A : A \otimes A^* \to I$ is the trace operator.

Rel is a *self-dual* compact closed category. That is, every object is its own dual $(X^* = X)$. The unit $\eta_X : I \to X \otimes X$ and counit $\epsilon_X : X \otimes X \to I$ are defined by

$$(*) \, \eta_X \, (x, x') \qquad \text{iff} \qquad x = x' \qquad \text{iff} \qquad (x, x') \, \epsilon_X \, (*).$$

In the graphical language, these morphisms are represented as "bent wires":

10

The zig-zag identities assert that "zig-zags can be straightened out":

$$
\vcenter{\hbox{}} \quad = \quad \vcenter{\hbox{}} \quad = \quad \vcenter{\hbox{}}.
$$
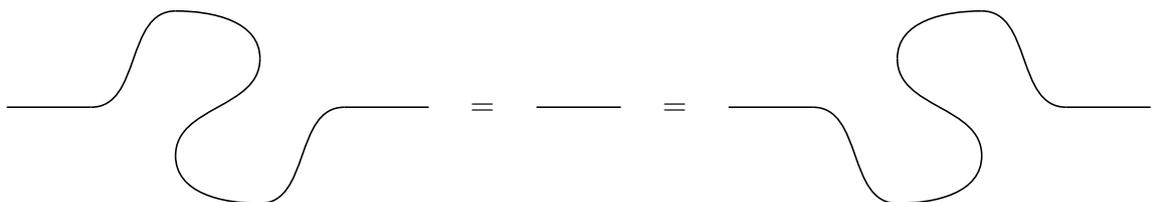
Remark. String diagrams for compact closed categories typically include arrowheads on the wires to distinguish objects $\xrightarrow{A}$ from their duals $\xrightarrow{A^*}$, which are drawn as reversed arrows $\xleftarrow{A}$. Because we work in self-dual categories, we can safely omit the arrowheads.

We have amassed enough structure to specify relations of arbitrary arity. Relations $B : I \to I$ whose domain and codomain are the monoidal unit constitute the degenerate case of arity zero. Since $I = \{*\}$ is the singleton set, the hom-set $\mathbf{Rel}(I, I)$ has only two members: the identity relation $1_I = \{(*, *)\}$ and the empty relation $\emptyset$. Defining $\top := 1_I$ and $\bot := \emptyset$, we interpret relations $B : I \to I$ as *booleans*.

Next, we have unary and binary relations. A unary relation $C : X \to I$ is called a *class* or *concept* in the description logic literature. Its elements have the form $(x, *)$, where $x \in X$. In the graphical calculus, wires of type I are not drawn at all, so a concept $C : X \to I$ is represented as

$$
\vcenter{\hbox{}}.
$$
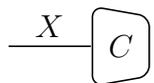
A binary relation $R : X \to Y$, or *role* in description logic jargon, is depicted as

$$
\vcenter{\hbox{}},
$$
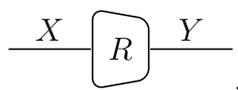
as we have seen.

Finally, we can easily express higher-order relations. A relation of arity n is a morphism of form $R : X_1 \otimes \cdots \otimes X_n \to I$. For instance, here is a ternary relation $R : X \otimes Y \otimes Z \to I$:
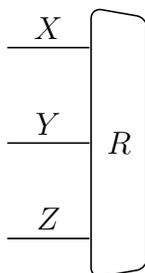
Apparently, there are two conventions for representing a binary relation: as a morphism $R : X \to Y$ or a morphism $R : X \otimes Y \to I$. By bending wires, we can pass freely between the two representations, via the transformations

and
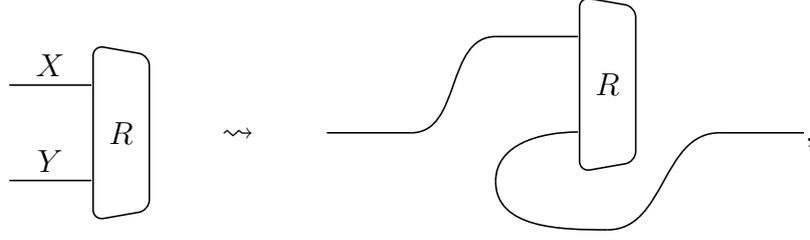
which, by the zig-zag identities, are mutually inverse. Most description logics do not support relations of arity greater than two, in part because the point-free textual syntax becomes quite awkward [Baa+07, §5.7]. The graphical language of monoidal categories enables graceful and intuitive composition even when relations have multiple inputs and outputs.

3.2. Dagger category

Every relation $R : X \to Y$ in **Rel** has an *opposite* relation $R^\dagger : Y \to X$, also known as the *converse* or *inverse* relation, defined by

$$y R^\dagger x \qquad \text{iff} \qquad x R y.$$

This structure is axiomatized by the following definition.

Definition. A *dagger category* is a category $\mathcal{C}$ equipped with a contravariant functor $(-)^\dagger : \mathcal{C}^{\mathrm{op}} \to \mathcal{C}$ that is the identity on objects and is involutive, i.e., $((-)^\dagger)^\dagger = 1_\mathcal{C}$.

More explicitly, a dagger category is a category $\mathcal{C}$ such that to every morphism $f : A \to B$ there corresponds a morphism $f^\dagger : B \to A$, and the correspondence satisfies

$$1_A{}^\dagger = 1_A, \qquad (fg)^\dagger = g^\dagger f^\dagger, \qquad \text{and} \qquad (f^\dagger)^\dagger = f.$$

When $\mathcal{C}$ is a monoidal category, one typically asks that the dagger respect the monoidal structure.

Definition. A symmetric monoidal category $\mathcal{C}$ is a *dagger symmetric monoidal category* if $\mathcal{C}$ is a dagger category and $(-)^\dagger$ is a symmetric monoidal functor, i.e.,

$$(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger \qquad \text{and} \qquad \sigma^\dagger_{A,B} = \sigma_{B,A} = \sigma^{-1}_{A,B}.$$

A compact closed category $\mathcal{C}$ is a *dagger compact category* if $\mathcal{C}$ is a dagger symmetric monoidal category and for each object $A \in \mathcal{C}$, there is a commutative diagram

$$
\begin{array}{ccc}
I & \xrightarrow{\;\epsilon_A^\dagger\;} & A \otimes A^* \\
& \eta_A \searrow & \downarrow{\scriptstyle \sigma_{A,A^*}} \\
& & A^* \otimes A
\end{array}
\; .
$$

Remark. Dagger compact categories have been introduced and studied in the context of quantum computing [Sel07]. The prototypical example is **Hilb**, the category of finite-dimensional Hilbert spaces and linear maps, equipped with the tensor product. Here $f^\dagger$ is the usual adjoint (or Hermitian conjugate) of the linear map f. For this reason, in a general dagger category $\mathcal{C}$, the morphism $f^\dagger$ is often called the *adjoint* of f. To avoid confusion with the 2-categorical notion of adjoint, invoked in Section 4, we do not use this terminology.

In the graphical language, $f^\dagger$ is represented by taking the "mirror image" of f:

$$\underset{B}{} \boxed{f^\dagger} \underset{A}{} \quad := \quad \underset{B}{} \boxed{f} \underset{A}{}.$$

Equivalently, given any morphism f drawn as a string diagram—which we read from left to right, as usual—we get $f^\dagger$ by reading the same diagram from right to left.

With the dagger operation defined above, **Rel** is a dagger compact category. However, unlike most dagger compact categories, **Rel** is self-dual. Thus there is potentially a second way to transform a morphism $X \to Y$ into a morphism $Y \to X$: bend both the input and the output wires. These two operations are actually the same, which we can express graphically as:

This equation is a generalization of the zig-zag identity: if we imagine "straightening out" the right-hand side by pulling on the ends of the input and output wires, we obtain the left-hand side. Mathematically speaking, the dagger structure on **Rel** is superfluous since it can be reduced to the compact closed structure. We choose to make the dagger structure explicit because inverse relations occur frequently in practice and the associated graphical language is succinct and intuitive.

3.3. Diagonals and codiagonals

In our final topic of this section, we show that the category of relations has operations for "copying" and "deleting" data and, dually, for "merging" and "creating" data. Using these operations, we can express intersections of classes and relations and, more generally, logical operations involving conjunction. We also obtain an important characterization of the "functional relations", or *maps*, in **Rel**.

The "copying" and "merging" operations are defined by internal comonoids and monoids, respectively, in **Rel**. We recall the general definition of a (co)monoid in a monoidal category.

Definition. Let $(\mathcal{C}, \otimes, I)$ be a monoidal category. An *internal monoid* in $\mathcal{C}$ is an object $M \in \mathcal{C}$ together with a *multiplication* morphism $\mu : M \otimes M \to M$ and a *unit* morphism $\eta : I \to M$ such that

$$
\begin{array}{ccc}
M \otimes M \otimes M & \xrightarrow{1_M \otimes \mu} & M \otimes M \\
\downarrow{\scriptstyle \mu \otimes 1_M} & & \downarrow{\scriptstyle \mu} \\
M \otimes M & \xrightarrow{\mu} & M
\end{array}
\qquad
\begin{array}{ccccc}
I \otimes M & \xrightarrow{\eta \otimes 1_M} & M \otimes M & \xleftarrow{1_M \otimes \eta} & M \otimes I \\
& \searrow & \downarrow{\scriptstyle \mu} & \swarrow & \\
& & M & &
\end{array}
$$

Dually, an *internal comonoid* in $\mathcal{C}$ is an internal monoid in $\mathcal{C}^{\mathrm{op}}$. In concrete terms, an internal comonoid is an object C together with a *comultiplication* morphism $\delta : C \to C \otimes C$ and a *counit* morphism $\epsilon : C \to I$ such that

$$
\begin{array}{ccc}
C & \xrightarrow{\delta} & C \otimes C \\
\downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \delta \otimes 1_C} \\
C \otimes C & \xrightarrow{1_C \otimes \delta} & C \otimes C \otimes C
\end{array}
\qquad
\begin{array}{ccccc}
& & C & & \\
& \swarrow & \downarrow{\scriptstyle \delta} & \searrow & \\
I \otimes C & \xleftarrow{1_C \otimes \epsilon} & C \otimes C & \xrightarrow{\epsilon \otimes 1_C} & C \otimes I
\end{array}
$$

If $\mathcal{C}$ is a symmetric monoidal category, we say that an internal monoid (M, μ, η) is *commutative* if

$$
\begin{array}{ccc}
M \otimes M & \xrightarrow{\sigma_{M,M}} & M \otimes M \\
& \searrow{\scriptstyle \mu} & \downarrow{\scriptstyle \mu} \\
& & M
\end{array}
$$

Dually, an internal comonoid (C, δ, ϵ) is *cocommutative* if

$$
\begin{array}{ccc}
C & \xrightarrow{\delta} & C \otimes C \\
& \searrow{\scriptstyle \delta} & \downarrow{\scriptstyle \sigma_{C,C}} \\
& & C \otimes C
\end{array}
$$

Note that an internal monoid in **Set** is just a monoid in the usual sense, i.e., a set M equipped an associative binary operation μ and an identity element η. Likewise, an internal commutative monoid in **Set** is just a commutative monoid.

We define a family of internal (co)monoids in **Rel** as follows. For each set X, define $\Delta_X : X \to X \otimes X$ by

$$x \, \Delta_X \, (x', x'') \qquad \text{iff} \qquad x = x' \wedge x = x'',$$

and define $\Diamond_X : X \to I$ by $\Diamond_X = \{(x, *) : x \in X\}$ (so that $x(\Diamond_X)*$ holds for every $x \in X$). It is easily verified that $(X, \Delta_X, \Diamond_X)$ is a cocommutative comonoid in **Rel**. By taking the opposite relations

$$\nabla_X := \Delta_X^\dagger : X \otimes X \to X \qquad \text{and} \qquad \Box_X := \Diamond_X^\dagger : I \to X,$$

we also obtain for each set X a commutative monoid $(X, \nabla_X, \Box_X)$ in **Rel**. We think of Δ_X as "copying" or "duplicating," ∇_X as "merging," $\Diamond_X$ as "deleting" or "erasing," and $\Box_X$ as "creating." These interpretations will be manifest from the graphical language, to be demonstrated shortly. However, for the graphical language to be consistent, the family of (co)monoids must satisfy certain *coherence axioms*, which ensure that they interact properly with the monoidal product. These axioms are captured by the following definition.

Definition ([Sel99])**.** A *monoidal category with diagonals* is a symmetric monoidal category $\mathcal{C}$ together with a family of morphisms $\Delta_A : A \to A \otimes A$ and $\Diamond_A : A \to I$, not necessarily natural in objects A, such that each triple $(A, \Delta_A, \Diamond_A)$ is a cocommutative comonoid in $\mathcal{C}$ and obeys the *coherence axioms*

$$\Diamond_I = 1_I, \qquad \Diamond_{A \otimes B} = \Diamond_A \otimes \Diamond_B, \qquad \Delta_{A \otimes B} = (\Delta_A \otimes \Delta_B)(1_A \otimes \sigma_{A,B} \otimes 1_B).$$

Dually, a *monoidal category with codiagonals* is a symmetric monoidal category $\mathcal{C}$ together with a family of morphisms $\nabla_A : A \otimes A \to A$ and $\square_A : I \to A$ such that each triple $(A, \nabla_A, \square_A)$ is commutative monoid in $\mathcal{C}$ and obeys the coherence axioms

$$\square_I = 1_I, \qquad \square_{A \otimes B} = \square_A \otimes \square_B, \qquad \nabla_{A \otimes B} = (1_A \otimes \sigma_{B,A} \otimes 1_B)(\nabla_A \otimes \nabla_B).$$

Remark. When the monoidal category $\mathcal{C}$ is, like **Rel**, not strict, we also need coherence axioms asserting that Δ_I and ∇_I are the unitors realizing the isomorphism $I \cong I \otimes I$.

The graphical language of a monoidal category with diagonals is

$$\Delta_A \quad = \quad \overset{A}{\underset{}{\rule{0pt}{0pt}}} \!\!\!\! , \qquad\qquad \Diamond_A \quad = \quad \overset{A}{\underset{}{\rule{0pt}{0pt}}}.$$

Similarly, the graphical language of a monoidal category with codiagonals is

$$\nabla_A \quad = \quad \overset{A}{\underset{}{\rule{0pt}{0pt}}} \!\!\!\! , \qquad\qquad \square_A \quad = \quad \overset{A}{\underset{}{\rule{0pt}{0pt}}}.$$

By the coherence axioms, we can express the diagonal morphisms for a product $A \otimes B$ in the graphical language as

$$\Delta_{A \otimes B} \quad = \quad , \qquad\qquad \Diamond_{A \otimes B} \quad = \quad .$$

Of course, there is a dual picture for the codiagonal morphisms.

Under the above definitions, **Rel** is a monoidal category with diagonals and codiagonals. A quick calculation shows that the intersection $R \cap S$ of two relations $R, S : X \to Y$ with common domain X and codomain Y is $\Delta_X (R \otimes S) \nabla_Y$ or, in graphical language,

$$R \cap S \quad = \quad .$$

As a special case, the intersection of two classes $C, D : X \to I$ is

$$C \cap D \quad = \quad \text{[string diagram: X branching to C and D]}.$$
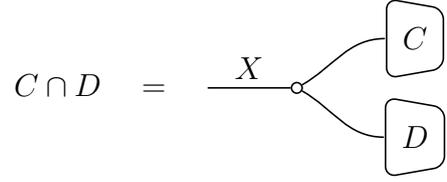
We can also express many of the basic concept constructors in description logic. For any relation $R : X \to Y$ and class $C : Y \to I$, the "limited" existential quantification $\exists R.\top$ is the class

$$\text{[string diagram: $X \;—\; R \;—\; Y \;\circ$]}$$

and the "full" existential quantification $\exists R.C$ is the class

$$\text{[string diagram: $X \;—\; R \;—\; Y \;—\; C$]}.$$
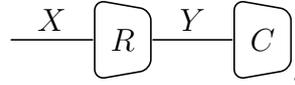
In contrast to description logic, we can retain access to the domain or codomain of the relation R while restricting its values. For instance, given classes $C : X \to I$ and $D : Y \to I$, the relation

$$\text{[string diagram: X branching through R, C, D to Y]}$$
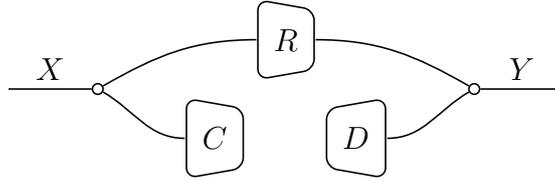
consists of all pairs $(x, y) \in X \times Y$ satisfying $xRy \wedge xC* \wedge yD*$. The value restriction concept constructor $\forall R.C$ cannot be expressed as a single morphism, but we can achieve the same effect by declaring a subsumption of two different morphisms:

$$\text{[string diagram: $\circ\;—\;X\;—\;R\;—\;Y$]} \quad \Longrightarrow \quad \text{[string diagram: $C\;—\;Y$]}.$$
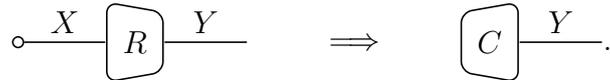
This 2-morphism asserts that $\forall x \in X.\forall y \in Y.(xRy \to yC*)$.

Finally, we can express a typed variant of the "universal role" in description logic. The local maximum

$$\top_{X,Y} \quad := \quad \text{[string diagram: $X\;\circ \qquad \circ\;Y$]}$$
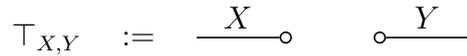
is the (unique) maximum element of the poset $\mathbf{Rel}(X, Y)$, namely $X \times Y$. It generalizes the top element $\top = \top_{I,I} = 1_I$ of the booleans $\mathbf{Rel}(I, I)$.

Maps   The diagonal structure on $\mathbf{Rel}$ leads to an abstract characterization of the relations that are functions, i.e., the relations $R : X \to Y$ with the property that for every $x \in X$, there exists a unique $y \in Y$ such that xRy. This matter is closely connected to the naturality, or lack thereof, of the diagonal in $\mathbf{Rel}$. In general, a diagonal in a symmetric

16

monoidal category $\mathcal{C}$ is *natural* if for every morphism $f : A \to B$, we have $f\Delta_B = \Delta_A(f \otimes f)$ and $f\Diamond_B = \Diamond_A$, or graphically

and

The first equation has the interpretation that applying f, then copying the output is the same as copying the input, then applying f to both copies; the second that applying f, then deleting the output is the same as deleting the input. When both equations hold for a morphism f, we say that f is a *comonoid homomorphism*. In a general category, we expect the equations to hold for morphisms that "behave like functions."

The diagonal in **Rel** is not natural because not all relations are functions. However, for any relation $R : X \to Y$, there are 2-morphisms

We say that every morphism in **Rel** is a *lax comonoid homomorphism*. Explicitly, the 2-morphisms are the inclusions

$$\{(x, y, y) : xRy\} \subseteq \{(x, y, y') : xRy \wedge xRy'\}$$
$$\{(x, *) : \exists y \in Y. xRy\} \subseteq \{(x, *) : x \in X\}.$$

When the first inclusion is an equality, R is a *partial function*; when the second is an equality, R is *total*; when both are equalities, R is a *function* or a *map*. In other words, the comonoid homomorphisms in **Rel** are exactly the relations that are functions.

Of course, for every concept about the diagonal, there is a dual concept about the codiagonal, whose details we omit. In **Rel**, we obtain abstract characterizations of the *injective*, *surjective*, and *bijective* relations. By combining the diagonal and codiagonal structures, we can characterize the injective functions, surjective functions, etc.

Interactions between structures   To conclude this section, we consider how the diagonals and codiagonals of **Rel** interact with each other and with the previous structures. In fact, the self-dual compact closed structure is reducible to the (co)diagonals. The unit and counit morphisms are given by

We have seen that the dagger is, in turn, reducible to the compact closed structure. Like the dagger operation, bending arrows is useful enough to merit its own textual and graphical syntax.

The internal monoids and comonoids in **Rel** combine to form internal *Frobenius algebras* (sometimes called *Frobenius monoids*) [Koc04]. That is, for each object X, there is a monoid $(X, \nabla_X, \square_X)$ and a comonoid $(X, \Delta_X, \Diamond_X)$ satisfying the *Frobenius equations*

$$(1_X \otimes \Delta_X)(\nabla_X \otimes 1_X) = \nabla_X \Delta_X = (\Delta_X \otimes 1_X)(1_X \otimes \nabla_X)$$

or, graphically,
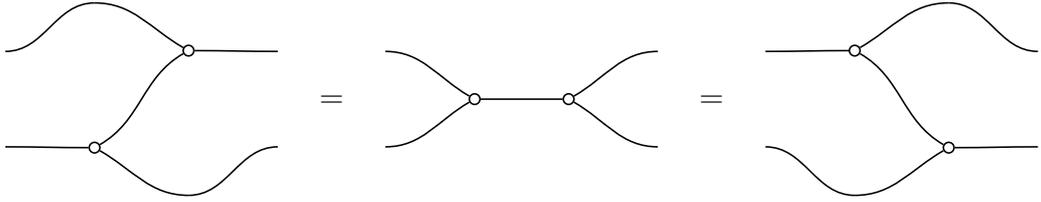
The monoid and comonoids are also *special*, meaning that $\Delta_X \nabla_X = 1_X$ or

Finally, by definition, we have $\nabla_X = \Delta_X^\dagger$ and $\square_x = \Diamond_X^\dagger$. These properties can be summarized by saying that $(X, \Delta_X, \Diamond_X, \nabla_X, \square_X)$ is a *special †-Frobenius monoid* [BE15].

4. Abstract categories of relations

The category **Rel** of sets and relations cannot stand alone as a formalism for knowledge representation. A knowledge representation system must be implementable on a computer, which requires that each knowledge base admit a finite description. Yet **Rel**, far from being a finitary object, has as objects every possible set and as morphisms every possible relation! Moreover, there is no formal system for specifying equations or subsumptions that should hold between relations. To enable a finite description of categories that "behave like" the category of sets and relations, we must *axiomatize* the salient structures of **Rel**. The previous section provides some clues about how to achieve this axiomatization.

In fact, there are two different notions of an "abstract" category of relations in the category theory literature. The best known is Freyd's *allegory*, popularized by Freyd and Scedrov [FS90] and utilized in Johnstone's treatise on topos theory [Joh02]. There have been a few efforts to apply allegories to real-world phenomena, e.g., in circuit design [BH94; BJ94], logic programming [AL12], and database modeling [ZMS13]. Allegories take intersections and the dagger (called "reciprocation") as primitive, characterizing the former by axioms like reflexivity, commutativity, and, most distinctively, the *modular law*

$$RS \cap T \subseteq (R \cap TS^\dagger)S,$$

where $R \subseteq S$ is, by definition, equivalent to $R \cap S = R$. (The reader can check that this rather strange law *does* hold in **Rel**.) The second notion is the *bicategory of relations*, introduced by Carboni and Walters [CW87; Car+08]. In bicategories of relations, the monoidal structures are primitive, while intersections and the dagger are derived concepts. The two notions are ostensibly quite different, but it can be shown that the categories of unitary pretabular allegories and of bicategories of relations are equivalent, in fact isomorphic [KN94; Law15]. Thus, the choice of axiomatization is mostly a matter of preference.

In this paper, we shall take bicategories of relations as our preferred notion of an "abstract" category of relations. An advantage of this choice is that the graphical language of monoidal categories is immediately available.

Definition ([CW87]). A *bicategory of relations* is a locally posetal 2-category $\mathcal{B}$ that is also a symmetric monoidal category $(\mathcal{B}, \otimes, I)$ with diagonals $(X, \Delta_X, \Diamond_X)_{X \in \mathcal{B}}$, such that
  • every morphism $R : X \to Y$ is a lax comonoid homomorphism,
$$R \cdot \Delta_Y \implies \Delta_X(R \otimes R), \qquad R \cdot \Diamond_Y \implies \Diamond_X;$$
  • the duplication morphisms Δ_X and deletion morphisms $\Diamond_X$ have right adjoints $\nabla_X := \Delta_X^*$ and $\Box_X := \Diamond_X^*$;
  • the pairs of morphisms (Δ_X, ∇_X) obey the Frobenius equations.

We denote by **BiRel** the category of (small) bicategories of relations and structure-preserving functors.

Remark. Our definition differs from Carboni and Walter's definition in one respect. They ask not for diagonals but only for internal cocommutative comonoids, subject to the requirement that they are the *unique* cocommutative comonoids with right adjoints. However, it appears that the only use of this uniqueness axiom is to derive the coherence axioms [CW87, Remark 1.3 (ii)]. We think it simpler to just assert the coherence axioms to begin with. By omitting the uniqueness axiom, we ensure that the theory of bicategories of relations is essentially algebraic (see below).

Every structure invoked in the definition has been introduced in Section 3, with the exception of adjoints. In this paper we use "adjoint" in the sense of 2-categories [Lac10]. Thus, in a locally posetal 2-category $\mathcal{C}$, a morphism $f : A \to B$ is *left adjoint* to $g : B \to A$ (and g is *right adjoint* to f), written $f \dashv g$, if
$$1_A \implies fg \qquad \text{and} \qquad gf \implies 1_B.$$
If a morphism f has a right adjoint g, then it is unique, for if g' is another right adjoint, then $g' \Rightarrow g'fg \Rightarrow g$ and, by symmetry, $g \Rightarrow g'$, so that $g = g'$. Similarly, left adjoints are unique when they exist. In **Rel**, a relation $R : X \to Y$ has a right adjoint $R^* : Y \to X$ if and only if R is a function, in which case $R^* = R^\dagger$. Together with the discussion in Section 3, this proves that **Rel** is a bicategory of relations. We shall meet other interesting bicategories of relations in Sections 6 and 8.

Carboni and Walters derive from the axioms of a bicategory of relations all the categorical structures discussed in Section 3. The situation is perfectly analogous to that of **Rel**. For the reader's convenience, we summarize the results in Table 2, using the textual syntax for brevity.

Structure	Name	Notation and definition
category	composition	$R \cdot S$
monoidal category	product	$R \otimes S$
	braiding	$\sigma_{X,Y}$
diagonal	copy	Δ_X
	delete	$\Diamond_X$
codiagonal	merge	$\nabla_X := \Delta_X^* = \Delta_X^\dagger$
	create	$\Box_X := \Diamond_X^* = \Diamond_X^\dagger$
compact closed	unit	$\eta_X := \Box_X \cdot \Delta_X$
	counit	$\epsilon_X := \nabla_X \cdot \Diamond_X$
dagger	dagger	$R^\dagger := (\eta_X \otimes 1_Y)(1_X \otimes R \otimes 1_Y)(1_X \otimes \epsilon_Y)$
logical	intersection	$R \cap S := \Delta_X (R \otimes S) \nabla_Y$
	true	$\top := \Diamond_I \cdot \Box_I = 1_I$
	local maximum	$\top_{X,Y} := \Diamond_X \cdot \Box_Y$

Table 2: Summary of morphisms in a bicategory of relations

The characterization of maps in **Rel** also generalizes to an arbitrary bicategory of relations. A morphism $R : X \to Y$ in a bicategory of relations $\mathcal{B}$ is a *map* if it has a right adjoint $R^* : Y \to X$. Equivalent conditions are that R is a comonoid homomorphism or that R is left adjoint to $R^\dagger$ [CW87, Lemma 2.5]. The collection of maps in $\mathcal{B}$ is closed under composition and monoidal products and hence forms a symmetric monoidal category, which we denote by $\mathrm{Map}(\mathcal{B})$. In the motivating example, $\mathrm{Map}(\mathbf{Rel}) = \mathbf{Set}$. The diagonal on $\mathcal{B}$ is natural when restricted to $\mathrm{Map}(\mathcal{B})$, making $\mathrm{Map}(\mathcal{B})$ into a cartesian category. In fact, $\mathrm{Map}(\mathcal{B})$ is the largest subcategory of $\mathcal{B}$ that is cartesian. Thus, in the terminology of [Sel99], $\mathrm{Map}(\mathcal{B})$ is the *focus* of $\mathcal{B}$.

5. Relational ologs

A categorical framework for knowledge representation, generalizing the category of sets and relations, emerges almost automatically from the abstractions developed in the previous section. An ontology in this framework is called a "relational olog," after Spivak and Kent [SK12]. We will define a relational olog to be any bicategory of relations that admits a finite description; more precisely, a relational olog is a *finitely presented* bicategory of relations. Intuitively, a finitely presented bicategory of relations is the "generic" or "free" bicategory of relations that contains a specified finite collection of basic objects, morphisms, and 2-morphisms. It is analogous to other free constructions in algebra, such as a free vector space or a finitely presented group. Another, more relevant example is a functional olog that does not involve limits or colimits, which is just a finitely presented category.

Definition. A *relational ontology log* (or *relational olog*) is a finitely presented bicategory of relations.

In more detail, a relational olog is a bicategory of relations $\mathcal{B}$ presented by
- a finite set of *basic types* or *object generators*;

- a finite set of *basic relations* or *morphism generators* of form $R : X \to Y$, where X, Y are object expressions;
- a finite set of *subsumption axioms* or *2-morphism generators* of form $R \Rightarrow S$, where R, S are well-formed morphism expressions with the same domain and codomain.

Note that while our definition does not explicitly include *equality axioms*, equality of morphisms can be reduced to subsumption. In the sequel, axioms of form $R = S$ are understood to be shorthand for the two axioms $R \Rightarrow S$ and $S \Rightarrow R$.

We hope that the meaning of the definition is intuitively clear but let us be somewhat more precise about our terminology. By "well-formed morphism expressions" we mean expressions constructed from the morphism generators and the syntax of bicategories of relations (see Table 2) such that domains and codomains are respected in all compositions. Similarly, "object expressions" are expressions constructed from the object generators and the syntax of monoidal categories ($\otimes$ and I). A bicategory of relations $\mathcal{B}$ is "presented by" a given collection of generators $\mathcal{B}_0$ if $\mathcal{B}$ contains (an isomorphic copy of) $\mathcal{B}_0$ and if for every other bicategory of relations $\mathcal{B}'$ containing $\mathcal{B}_0$, there exists a unique functor $F : \mathcal{B} \to \mathcal{B}'$ preserving the structure of **BiRel** and the generators $\mathcal{B}_0$. As usual, this universal property guarantees the *uniqueness* of $\mathcal{B}$ up to isomorphism. For readers concerned about the *existence* of $\mathcal{B}$ we make the following technical remark.

Remark. The preceding definition can be made fully rigorous by formulating the axioms of a bicategory of relations as an essentially algebraic theory. Roughly speaking, an *essentially algebraic theory* is an algebraic theory that allows some operations to be partially defined, provided the domain of definition is characterized by equations between total operations [Fre72; PS97]. A motivating example is the theory of categories, where composition of morphisms is partially defined. It is well known that definition by finite presentation, also known as the "method of generators and relations," works in any essentially algebraic theory. There are several methods for constructing a free model from the syntax of the theory. *Generalized algebraic theories*, a reformulation of essentially algebraic theories using dependent type theory, provide a particularly elegant solution [Car86; Pit95].

Alternatively, our foray into categorical logic (Section 8) yields an entirely different and fully explicit construction of relational ologs. This construction is based not on universal algebra or dependent type theory but on the proof theory of a subset of first-order logic.

5.1. Example: Friend of a friend

While the formal definition of a relational olog is somewhat abstract, the specification of a particular relational olog is, as a practical matter, simple and intuitive, thanks to the graphical language of monoidal categories. To illustrate, we specify a relational olog in a toy domain affectionately called "friend of a friend" (or "FOAF") [BM14; DV10]. This domain, involving people, organizations, and their presences online, is often used to showcase the Semantic Web technologies (RDF and OWL). We take the formal specification of FOAF as an inspiration only, making no attempt to replicate its interface or general philosophy.

The basic types of the olog are "Person", "Organization", "Number", and "String". We shall introduce the basic relations as we need them. Here are some essential relations for

our ontology:

Person — [knows] — Person , Person — [member of] — Organization ,

Person — [friend of] — Person , Person — [works at] — Organization .
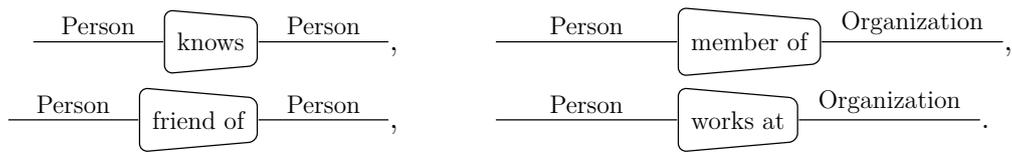
The obligatory "friend of a friend" relation is just the composite
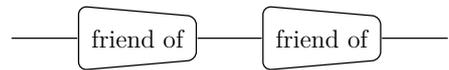
— [friend of] — [friend of] — .

(All relations are typed but when the types are clear from context we shall suppress the type labels.) Some of the basic relations are subsumed by others. For instance, if Alice is a friend of Bob, then Alice knows Bob; thus, the "friend of" relation is subsumed by the "knows" relation:

— [friend of] — $\implies$ — [knows] — .
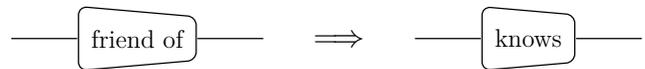
Likewise, the "works at" relation is subsumed by the "member of" relation (diagram omitted). Presumably, if Alice knows Bob, then Bob also knows Alice, so we should declare that the "knows" relation is *symmetric*:

— [knows] — $=$ — [knows] — .

Most people would also say that the "friend of" relation is symmetric.

We can attach some basic data to each person, such as their name and age:

Person — [age] — Number ,

Person — [family name] — String , Person — [given name] — String .
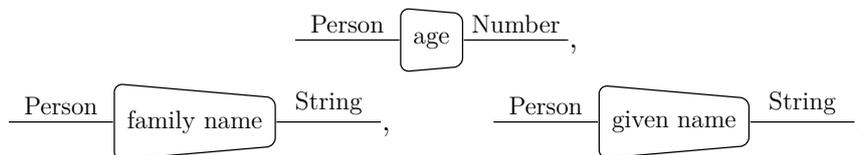
We declare that these relations are (total) *functions*; for instance,

— [age] —< $=$ —<[age] , [age] —<[age]> , — [age] —∘ $=$ —∘.
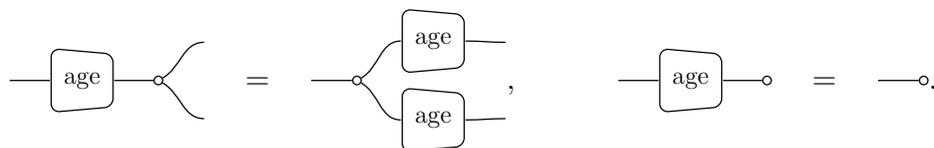
In RDF and OWL, functional relations whose codomains are primitive data types are called "properties" and are treated specially.

For extra flavor, we complement the "friend of" relation with an "enemy of" relation. We can then define the notorious relation of "frenemy" as the *intersection* of friend and enemy:

— [frenemy of] — $:=$ —∘< [friend of] , [enemy of] >∘— .
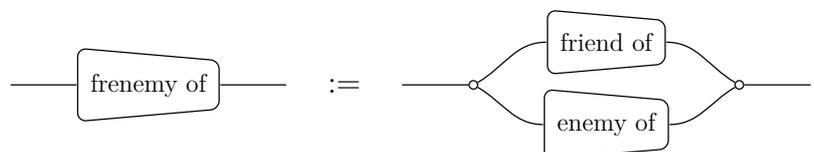
Next, we model some basic family relationships. Having introduced a "child of" relation, the "parent of" relation is just its *inverse*:

$$\boxed{\text{parent of}} \quad := \quad \boxed{\text{child of}}.$$
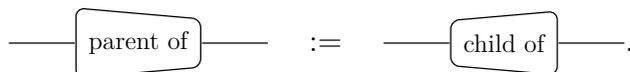
An "ancestor of" relation should possess several properties. First, it should subsume "parent of":

$$\boxed{\text{parent of}} \quad \implies \quad \boxed{\text{ancestor of}}.$$
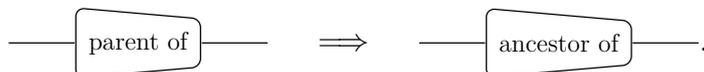
It should be *transitive*,

$$\boxed{\text{ancestor of}}\ \boxed{\text{ancestor of}} \quad \implies \quad \boxed{\text{ancestor of}},$$

(an ancestor of an ancestor is an ancestor) and *reflexive*,

$$\underline{\text{Person}} \quad \implies \quad \boxed{\text{ancestor of}},$$

(by convention, we regard every person as their own ancestor). Finally, the ancestor relation should be *antisymmetric*,

$$\overset{\boxed{\text{ancestor of}}}{\underset{\boxed{\text{ancestor of}}}{\circ\qquad\circ}} \quad \implies \quad \underline{\text{Person}},$$
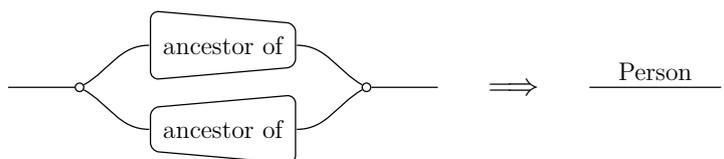
(if two people are both ancestors and descendants of each other, then they are the same person). We can now *deduce*, rather an declare as an axiom, that the relation "grandparent of" is subsumed by "ancestor of":

$$\boxed{\text{grandparent of}} \quad := \quad \boxed{\text{parent of}}\ \boxed{\text{parent of}}$$

$$\implies \quad \boxed{\text{ancestor of}}\ \boxed{\text{ancestor of}}$$
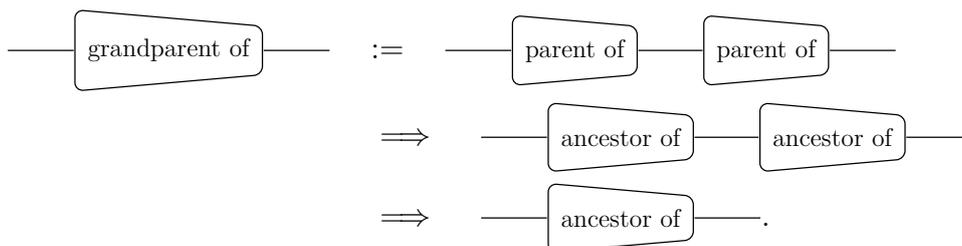
$$\implies \quad \boxed{\text{ancestor of}}.$$

In summary, "ancestor of" is a partial order that subsumes "parent of." It would be more precise to declare that "ancestor of" is the partial order *generated* by "parent of," but that cannot be expressed in a relational olog.

So far we have seen only relatively simple, binary relations. Let us now consider more complex compound relations and relations of arity different than two. The class (unary relation) of employed people can be defined as the class of people who work at some organization:

$$\underline{\text{Person}}\ \boxed{\text{is employed}} \quad := \quad \underline{\text{Person}}\ \boxed{\text{works at}}\ \overset{\text{Organization}}{}\circ.$$

23

We declare a ternary relation "salary" with signature

.
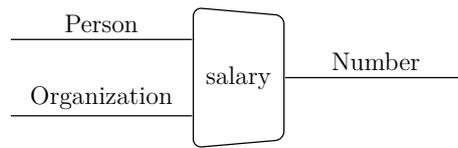
We assert that "salary" is a *partial function* (diagram omitted). Its domain of definition is characterized by

.
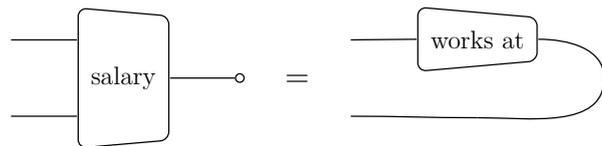
Alternatively, we can take this equation as the definition of the "works at" relation: a person works at an organization if and only if they draw a salary from that organization. As another example, a "colleague" is a person whom you know and with whom you share a membership at some organization:

.
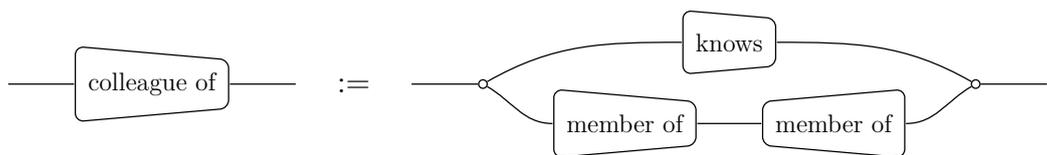
A simple calculation, using the symmetry of "knows," proves that "colleague of" is a symmetric relation. More fancifully, a romantic "love triangle" is the ternary relation

.
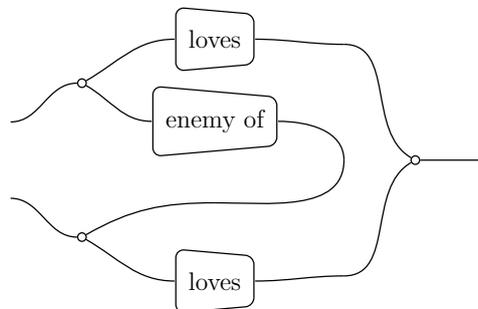
Thus, a love triangle consists of two people, mutually enemies, who both love a third person. Assuming "enemy of" is symmetric, this relation is symmetric in its first two arguments. As long as we're indulging in Shakespearean themes, we can also define the quaternary relation of "intergenerational family feud":

.
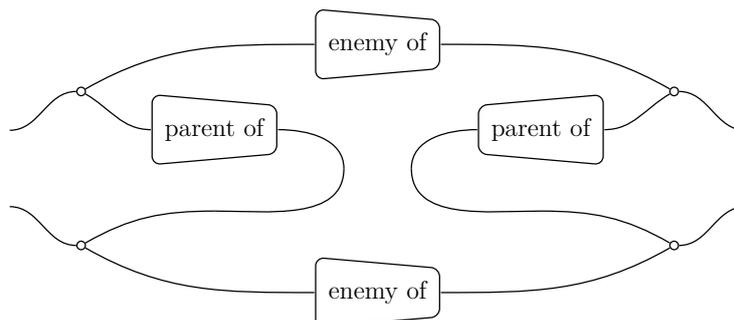
Such a feud consists of two parent-child pairs, where the parents are enemies and the children are also enemies.

Although it might be entertaining to continue along these lines, we shall stop here. We hope we have convinced the reader that relational ologs are both expressive and intuitive. With a little practice, it becomes easy to write down complex relations and read them at a glance. However, there are certain natural constraints that cannot be expressed in a relational olog, as developed so far. For instance, if we took our ontology more seriously, we might prefer to dismiss the possibility of "frenemies" and declare that "friend of" and "enemy of" are disjoint relations. At present we cannot express this constraint because we cannot express the empty relation. Nor can we express unions, so we cannot declare that, for example, the "parent of" relation is the union of the "mother of" and "father of" relations. In Section 9 we explain how to overcome these limitations.

6. Instance data

A distinguishing feature of categorical knowledge representation, compared to the logical paradigm, is a rich and flexible notion of instance data. The idea of instance data is simply that of *functorality*. To be precise, *instance data* for a relational olog $\mathcal{B}$ in an arbitrary bicategory of relations $\mathcal{D}$ is a structure-preserving functor $D : \mathcal{B} \to \mathcal{D}$. We call $\mathcal{D}$ the *data category* for the instance data D. Unsurprisingly, the "standard" data category is **Rel**, the category of sets and relations. We study this important case and several others below.

In knowledge representation systems based on description logic, instances are represented by named constants *within the logical system*, usually called "individuals." There are several advantages to the categorical notion of instance data. First, there is a clean separation between universal concepts, stored in the olog $\mathcal{B}$, and instantiations of these concepts, stored in the functor $D : \mathcal{B} \to \mathcal{D}$. In description logic, this separation is only partly achieved by partitioning the axioms of the knowledge base into a "TBox" and an "ABox" (see Section 2). Besides its aesthetic appeal, the separation of universal and particular knowledge has important practical benefits. In modern "big data" applications involving a large number of individuals, storing instance data in a suitable database, rather than as logical sentences, becomes a practical necessity. Of course, one can define ad hoc schemes for translating between the logical system and the database system. The point is that functors provide a simple, mathematically precise notion of "translation" between systems.

Another advantage, less easily achieved by ad hoc devices, is that we can define "non-standard" instance data by using data categories besides **Rel**. This possibility arises because ologs, unlike logical theories, are algebraic structures and hence come equipped with a general notion of structure-preserving maps, namely functors. From this point of view, instance data for relational ologs is closely connected to *functorial semantics* in categorical logic. We shall return to categorical logic in Section 8.

Let us add, parenthetically, that it *is* possible to represent individuals inside a relational olog. An *individual* of type X is a map $c : I \to X$, since a function from the singleton set $I = \{*\}$ to X picks out an element of X. In our view, individuals should be used

sparingly to represent concepts that are inherently singletons. For example, there is at any given time only one Dalai Lama, so it would be reasonable to represent the Dalai Lama as an individual of type "Person." It should suffice to include most "ordinary" people only as instance data. In general, the olog should contain only universal concepts, even if they are singletons, while the instance data contains all particular knowledge. (We grant that when building an ontology it is not always easy to distinguish between universal and particular, but often the difference is clear enough.)

In this section, we consider four different kinds of instance data for relational ologs. The first and second are interpreted as relational databases and graph databases, respectively. With minor modification, these two concepts apply equally well to functional ologs and are treated by Spivak and Kent [SK12]. The other two kinds of instance data are specific to relational ologs. From the matrix calculus of relations, an extension of the familiar boolean algebra, we derive instance data in the category of boolean matrices. Finally, we consider "non-standard" instance data in the category of linear relations. This data category can be used to model linear dynamical systems.

6.1. Relational databases

The default category for instance data, suitable for most applications, is the category of sets and relations. Thus, without further qualification, *instance data* for a relational olog $\mathcal{B}$ is a structure-preserving functor $D : \mathcal{B} \to \mathbf{Rel}$.

A structure-preserving functor $D : \mathcal{B} \to \mathbf{Rel}$ is defined by the following data. Each basic type X of $\mathcal{B}$ is mapped to a set $D(X)$ and each basic relation $R : X \to Y$ of $\mathcal{B}$ is mapped to a subset $D(R)$ of $D(X) \times D(Y)$. The set $D(X)$ contains the *instances* of type X and the subset $D(R)$ tabulates the instances of type X that are in relation R with instances of type Y. By functorality, this data determines the action of D on every object and every morphism of $\mathcal{B}$, since $\mathcal{B}$ is generated by the basic types and relations. Moreover, in order for D to be well-defined, it must preserve all the subsumption axioms of $\mathcal{B}$. Thus, for every subsumption axiom $R \Rightarrow S$, we require that $D(R) \subseteq D(S)$. If the mapping D satisfies these properties, it defines valid instance data for the olog.

Instance data is straightforwardly interpreted as a relational (SQL) database [Cod70]. In the idealized database interpretation, there is a single-column table $D(X)$ for each basic type X, which defines the primary key of each instance of type X, and a multi-column table $D(R)$ for each basic relation R, whose columns are foreign keys associated with the domain and codomain types of R. The tables $D(R)$ are called "association tables" or "junction tables" in SQL jargon. Association tables are the standard way of representing many-to-many relationships in a relational database. The primary key of the association table is the product of the foreign keys of the columns.

An example should make this clear. Instance data for a fragment of the "friend of a friend" ontology (Section 5.1) is shown below.

Person

ID
P1
P2
P3
P4

Organization

ID
O1
O2

salary

Person	Organization	Number
P1	O1	30,000
P3	O2	40,000

friend of

Person 1	Person 2
P1	P2
P2	P1

knows

Person 1	Person 2
P1	P2
P2	P1
P3	P4
P4	P3

As required by functorality, the "friend of" and "knows" tables are symmetric and the "friend of" table is a subset of the "knows" table. In practice we expect to deviate slightly from the idealized database interpretation to obtain a more compact database schema. Most importantly, instead of representing *maps* as individual tables, e.g.,

Person

ID
P1
P2
P3
P4

age

Person	Number
P1	21
P2	37
P3	22
P4	54

family name

Person	String
P1	Doe
P2	Smith
P3	Williams
P4	Jones

given name

Person	String
P1	Alice
P2	Bob
P3	Carol
P4	David

,

it would be more conventional to combine the maps with common domain into a single table, e.g.,

Person

ID	age	family name	given name
P1	21	Doe	Alice
P2	37	Smith	Bob
P3	22	Williams	Carol
P4	54	Jones	David

.

We could represent partial maps similarly, using `NULL` to indicate undefined values.

By formalizing the association of instance data with an olog, it becomes possible to migrate data in a precise, principled way. Suppose that, in light of new information or a changing world, we decide to update the concepts in our ontology $\mathcal{B}$, yielding a new ontology $\mathcal{B}'$. Ideally we can translate the concepts of $\mathcal{B}$ into concepts of $\mathcal{B}'$ by means of a functor $F : \mathcal{B} \to \mathcal{B}'$. The functor F can then be used to migrate the original data $D : \mathcal{B} \to \mathbf{Rel}$ to updated data $D' : \mathcal{B}' \to \mathbf{Rel}$. This paradigm is called *functorial data migration* and is investigated by Spivak and collaborators in a series of papers [Spi12; SK12; SW15; Sch+16]. Functorial data migration has been developed for functional ologs. At least one data migration functor, the pullback functor, has an obvious analogue for relational ologs. It is an open question whether the other data migration functors, the left and right pushforward functors, admit analogues. This question, while important, is not pursued further here.

6.2. Graph databases

Graph databases [AG08; RN10] provide a natural storage model for the instance data of an ontology. In fact, the development of graph databases can be traced back to the semantic networks and frame systems of the early era of knowledge representation [AG08, Fig. 1]. Even today it is sometimes suggested that graph databases *are* knowledge representation systems. That is not so: graph databases offer a generic data storage model that need not impose any logical constraints on the data. Still, the confusion exists precisely because graph databases are so well-aligned with the practice of knowledge representation. In this section, we explain how instance data $D : \mathcal{B} \to \mathbf{Rel}$ can be interpreted as a graph database.

Unlike relational databases, which are practically synonymous with the Structured Query Language (SQL), graph databases are an emerging technology with no universally accepted data model or query language. Our construction maps easily onto Apache TinkerPop3 [Apa15], an open standard for graph databases with moderate vendor adoption. The Resource Description Framework (RDF), a core component of the Semantic Web, can also be regarded as a graph database [AG05], especially when coupled with a graph query language like SPARQL [PS08].

The interpretation of instance data as a graph database involves a construction called the "category of elements."

Definition. Let $\mathcal{B}$ be a bicategory of relations. The *category of elements* of a structure-preserving functor $F : \mathcal{B} \to \mathbf{Rel}$, denoted $\int_{\mathcal{B}} F$ or $\int F$, has as objects, the pairs

$$(X, x) \qquad \text{where} \qquad X \in \mathcal{B}, \quad x \in F(X),$$

and as morphisms $(X, x) \to (Y, y)$, the morphisms in $\mathcal{B}$

$$R : X \to Y \qquad \text{such that} \qquad x \, F(R) \, y.$$

Composition and identity morphisms are inherited from $\mathcal{B}$.

Remark. Technically, this definition is not included in the usual notion of a *category of elements* [Rie16, §2.4], which applies to functors $F : \mathcal{C} \to \mathbf{Set}$, nor in the more general *Grothendieck construction* [Jac99, §1.10], which applies to functors $F : \mathcal{C} \to \mathbf{Cat}$. However, it is evidently the same idea, so we will use the same terminology.

A category of elements $\int_{\mathcal{B}} F$ is itself a bicategory of relations, with its structure inherited from both $\mathcal{B}$ and $\mathbf{Rel}$. The monoidal product is defined by

$$(X, x) \otimes (Y, y) := (X \otimes Y, (x, y)), \qquad I_{\int F} := (I_{\mathcal{B}}, *).$$

The diagonal maps are $\Delta_{(X,x)} := \Delta_X$ and $\Diamond_{(X,x)} := \Diamond_X$. These morphisms behave as expected because F is structure-preserving. For example, there is a copying morphism $\Delta_X : (X, x) \to (X \otimes X, (x', x''))$ in $\int F$ if and only if $x = x'$ and $x = x''$. Finally, the 2-morphisms of $\int F$ are just the 2-morphisms of $\mathcal{B}$. (It is tempting to declare that $R \Rightarrow S$ in $\int F$ whenever $F(R) \Rightarrow F(S)$ in $\mathbf{Rel}$, but under that definition $\int F$ is not necessarily locally posetal.)

Given instance data $D : \mathcal{B} \to \textbf{Rel}$, we think of $\int D$ as a graph database as follows. The vertices of the graph are the objects (X, x) of $\int D$. The vertex labels (or vertex types) are the objects X of $\mathcal{B}$, given by the canonical projection functor $\int_{\mathcal{B}} D \to \mathcal{B}$. The directed edges of the graph are morphisms $(X, x) \xrightarrow{R} (Y, y)$ of $\int D$. The edge labels (or edge types) are the morphisms R of $\mathcal{B}$, again given by the projection functor $\int_{\mathcal{B}} D \to \mathcal{B}$. As an example, the instance data for the FOAF ontology yields the graph database:
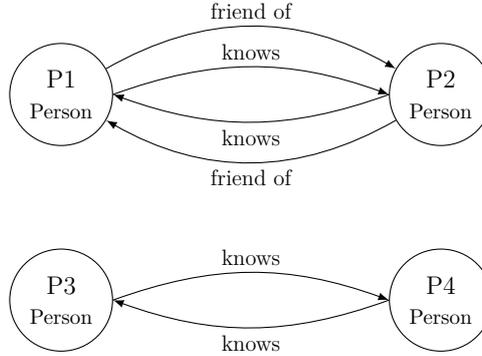
As with relational databases, the idealized graph database interpretation may require modification to accommodate real-world database systems. The size of the graph can be considerably reduced by representing maps with "primitive type" codomains (e.g., "Number" or "String") as "vertex properties," a feature supported by most graph databases. Symmetric relations, such as "knows" and "friend of," can be represented by one undirected edge instead of two directed edges. Another issue, not arising with relational databases, is the representation of relations whose domain or codomain is a product of basic types, such as the "salary" relation. If the database included vertices only for basic types, we would need directed *hyper*edges [Gal+93], a feature not supported by most graph databases. The solution is to include vertices for product types and edges for the projection morphisms. In fact, this encoding is accomplished automatically by the monoidal product in the category of elements.

6.3. Boolean matrices

We now consider instance data derived from the matrix calculus of relations. Unlike relational and graph databases, the matrix calculus has no analogue for functional ologs. It is a special case of *categorical matrix calculus*, which can be performed in any biproduct category [CP10, §3.5.5], [Har09].

Let $\mathbb{B} = \{0, 1\}$ be the commutative "rig" (commutative ring without negatives) of booleans, whose operations are defined by

$$0 + 0 = 0, \quad 0 + 1 = 1 + 0 = 1, \quad 1 + 1 = 1$$
$$0 \cdot 0 = 0, \quad 0 \cdot 1 = 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1.$$

That is, addition in $\mathbb{B}$ is logical disjunction and multiplication in $\mathbb{B}$ is logical conjunction.

Definition. The *category* $\textbf{Mat}(\mathbb{B})$ *of boolean matrices* has as objects the natural numbers and as morphisms $m \to n$ the $m \times n$ matrices over $\mathbb{B}$. Composition is defined by matrix multiplication and the identity morphisms are the identity matrices.

We interpret a matrix $R \in \mathbb{B}^{m \times n}$ as a relation with domain $[m] = \{1, \ldots, m\}$ and codomain $[n] = \{1, \ldots, n\}$, where individual $i \in [m]$ is in relation R with individual $j \in [n]$ if and only if $R_{i,j} = 1$. As expected, composition in $\mathbf{Mat}(\mathbb{B})$ is given by existential quantification:

$$(R \cdot S)_{i,k} = 1 \qquad \text{iff} \qquad \exists j : R_{i,j} = 1 \wedge S_{j,k} = 1.$$

The category of boolean matrices is a bicategory of relations. There is a 2-morphism $R \Rightarrow S$ if and only if $R \leq S$ (elementwise). The monoidal product is the tensor product of matrices:

$$R \otimes S := \begin{pmatrix} R_{1,1}S & \cdots & R_{1,n}S \\ \vdots & \ddots & \vdots \\ R_{m,1}S & \cdots & R_{m,n}S \end{pmatrix}.$$

The diagonals are defined by

$$\Delta_n := \begin{pmatrix} e_1 e_1^\top & \cdots & e_n e_n^\top \end{pmatrix} \in \mathbb{B}^{n \times n^2} \qquad \text{and} \qquad \Diamond_n := \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{B}^{n \times 1},$$

where e_i is the ith standard basis vector. With these definitions, the dagger is simply the matrix transpose, $R^\dagger = R^\top$. Given matrices $R, S \in \mathbb{B}^{m,n}$, a quick calculation shows that local intersections are given by the elementwise (Hadamard) product:

$$R \cap S := \Delta_m (R \otimes S) \nabla_n = \begin{pmatrix} R_{1,1}S_{1,1} & \cdots & R_{1,n}S_{1,n} \\ \vdots & \ddots & \vdots \\ R_{m,1}S_{m,1} & \cdots & R_{m,n}S_{m,n} \end{pmatrix} =: R \odot S.$$

Thus we recover the usual intersection of relations.

Anticipating Section 9, we equip $\mathbf{Mat}(\mathbb{B})$ with a *second* monoidal product, the direct sum of matrices:

$$R \oplus S := \begin{pmatrix} R & 0 \\ 0 & S \end{pmatrix}.$$

Define a codiagonal with respect to the direct sum by

$$\blacktriangledown_n := \begin{pmatrix} I_n \\ I_n \end{pmatrix} \in \mathbb{B}^{2n \times n} \qquad \text{and} \qquad \blacksquare_n := () = 0 \times n \text{ matrix},$$

where I_n is the $n \times n$ identity matrix. We recover unions of relations from the formula

$$R \cup S := \blacktriangle_m (R \oplus S) \blacktriangledown_n = R + S.$$

This construction will be revisited and generalized in Section 9.

Matrix data—instance data in the category of boolean matrices—for a relational olog $\mathcal{B}$ is a structure-preserving functor $D : \mathcal{B} \to \mathbf{Mat}(\mathbb{B})$. Each basic type X of $\mathcal{B}$ is mapped to a natural number $D(X)$ and each basic relation $R : X \to Y$ is mapped to a $D(X) \times D(Y)$ matrix $D(R)$ over $\mathbb{B}$, such that all the subsumption axioms of $\mathcal{B}$ are satisfied. For example, the instance data for the FOAF ontology becomes

$$D(\text{friend of}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad D(\text{knows}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The two matrices are symmetric because the corresponding relations in $\mathcal{B}$ are. Note that relations with (theoretically) infinite domain or codomain, such as "family name," "given name," and "salary," cannot be represented as matrices.

The matrix representation of relations plays an important role in data analysis applications. A symmetric relation R with the same domain and codomain, such as the "friend of" relation, is often regarded as an undirected graph, with $D(R)$ its *adjacency matrix*. This simple observation is the starting point of the spectral analysis of network data, a rich and active area of statistical research [Mah16]. Matrix data offers yet another example of how instance data can be used to connect an ontology to another computational system in a mathematically precise way.

6.4. Linear relations

Relational and graph databases are both realized by functors $D : \mathcal{B} \to \mathbf{Rel}$, and matrix data can be regarded as a repackaged functor $D : \mathcal{B} \to \mathbf{FinRel}$, where $\mathbf{FinRel}$ is the category of *finite* sets and relations. On the basis of these examples one might suppose that, in general, instance data amounts to a functor into $\mathbf{Rel}$. That would be mistaken. In this section, we describe a counterexample of practical significance, the category of *linear* relations. This category has been studied by Baez and Erbele [BE15], and independently by Bonchi, Sobociński, and Zanasi [BSZ14], as a model of signal flow diagrams in control theory. Note that the material in this section is peripheral to the main development of the paper and can be skipped without loss of continuity.

Definition. The *category of linear relations*, denoted $\mathbf{VectRel}_k$, is the category whose objects are finite-dimensional vector spaces (over a fixed field k) and whose morphisms $L : U \to V$ are *linear relations*, which are vector subspaces

$$L \subseteq U \oplus V.$$

Composition and identity morphisms are defined as in $\mathbf{Rel}$; thus, given linear relations $L : U \to V$ and $M : V \to W$, the composite $LM : U \to W$ is

$$LM = \{(u, w) \mid \exists v \in V : (u, v) \in L \land (v, w) \in M\}.$$

The category of linear relations is a bicategory of relations. The 2-morphisms are subspace inclusions. The monoidal product is the direct sum (which we always write as $\oplus$, not $\otimes$) and the monoidal unit is the zero vector space. The diagonal is defined by

$$\Delta_V : \begin{cases} V \to V \oplus V \\ v \mapsto (v, v) \end{cases} \qquad \text{and} \qquad \Diamond_V : \begin{cases} V \to \{0\} \\ v \mapsto 0. \end{cases}$$

Given these definitions, the maps in $\mathbf{VectRel}_k$ are just linear maps, in the usual sense. Be warned that the dagger is *not* the matrix transpose, in contrast to $\mathbf{Mat}(\mathbb{B})$; indeed, the transpose of a linear map is always another linear map, but $f^\dagger$ is not a linear map unless f is invertible. As in $\mathbf{Rel}$, the dagger simply effects a formal exchange of inputs and outputs.

However, the linear transpose leads to interesting structure not present in **Rel**. The transpose of the duplication map Δ_V is the *addition* map

$$\blacktriangledown_V := \Delta_V^\top : \begin{cases} V \oplus V \to V \\ (v_1, v_2) \mapsto v_1 + v_2 \end{cases}$$

and the transpose of the deletion map $\lozenge_V$ is the *zero* map

$$\blacksquare_V := \lozenge_V^\top : \begin{cases} \{0\} \to V \\ \quad 0 \mapsto 0. \end{cases}$$

The linear relations $\blacktriangle_V := \blacktriangledown_V^\dagger : V \to V \oplus V$ and $\blacklozenge_V := \blacksquare_V^\dagger : V \to \{0\}$ are called *coaddition* and *cozero*, respectively.

The family of maps $(\blacktriangledown_V, \blacksquare_V)$ form a codiagonal structure on $\mathbf{VectRel}_k$. Moreover, every linear relation $L : U \to V$ is a *lax monoid homomorphism* with respect to this structure, meaning that

$$(L \oplus L)\blacktriangledown_V \implies \blacktriangledown_U L \qquad \text{and} \qquad \blacksquare_V \implies \blacksquare_U L.$$

The duality exhibited here motivates the following definition.

Definition ([CW87, §5]). An *abelian bicategory of relations* is locally posetal 2-category $\mathcal{B}$ that is also a symmetric monoidal category $(\mathcal{B}, \otimes, I)$ with diagonals $(X, \Delta_X, \lozenge_X)$ and codiagonals $(X, \blacktriangledown_X, \blacktriangle_X)$ such that
- every morphism $R : X \to Y$ is a lax comonoid homomorphism and a lax monoid homomorphism;
- the morphisms $\Delta_X, \lozenge_X, \blacktriangledown_X, \blacksquare_X$ have right adjoints, denoted $\nabla_X, \square_X, \blacktriangle_X, \blacklozenge_X$;
- both pairs (Δ_X, ∇_X) and $(\blacktriangle_X, \blacktriangledown_X)$ obey the Frobenius equations.

Remark. An equivalent, more succinct definition is that an *abelian bicategory of relations* is a locally posetal 2-category $\mathcal{B}$ such that both $\mathcal{B}$ and $\mathcal{B}^{\mathrm{co}}$ are bicategories of relations with respect to the same monoidal product. Here $\mathcal{B}^{\mathrm{co}}$ is the 2-category $\mathcal{B}$ with all 2-morphisms reversed. See Section 9 for further discussion.

The category of linear relations, $\mathbf{VectRel}_k$ is an abelian bicategory of relations. The usual category of relations, **Rel**, is *not* an abelian bicategory of relations.

The category of linear relations would obviously be an inappropriate data category for the FOAF ontology. Linear relations are useful for representing systems of linear ordinary differential equations (ODEs), as argued by Baez and Erbele [BE15]. The graphical language of monoidal categories then formalizes the signal flow diagrams that appear in control theory and other engineering fields. In this setting, one takes the field $k = \mathbb{R}(s)$, the real numbers $\mathbb{R}$ with a formally adjoined indeterminate s. Upon taking Laplace transforms, differentiation becomes the linear operation of scalar multiplication by s and integration becomes multiplication by $1/s$. A linear relation $L : k^m \to k^n$ is a system of linear, constant-coefficient ODEs with m input signals and n output signals.

The damped, driven harmonic oscillator provides a simple, one-dimensional example. The equation of motion is

$$m\frac{d^2x}{dt^2} + \beta\frac{dx}{dt} + \kappa x(t) = F(t),$$

where x is a position or angle and F is a driving force. Provided the oscillations are not too large, this equation accurately describes a mass on a spring or a pendulum under gravity, subject to an additional driving force. The system can be represented by a linear relation $k \to k$ with input F and output x:
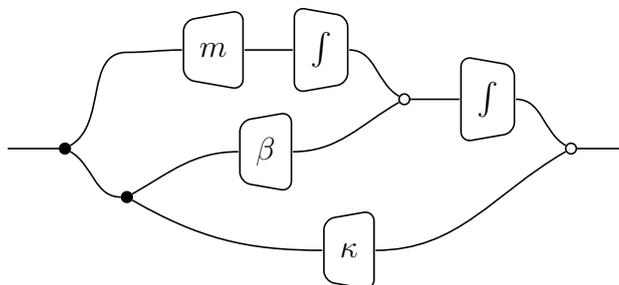
The dark nodes are coaddition and the light nodes are coduplication. It is perhaps easier to read the diagram from right to left, noting that the formal inverse of the integration map $\int$ is the differentiation map $\frac{d}{dx}$. The same diagram is drawn in conventional engineering notation in [Fri12, Fig. 2.1]. A string diagram for a more complicated system, the "inverted pendulum," is presented in [BE15, §5].

Like the previous data categories, the category of linear relations deserves a more comprehensive treatment than space here permits. We include it mainly as a concrete example of the "non-standard" instance data enabled by functorial semantics. The possibility of instance data with extra algebraic or topological structure is a distinctive—and sometimes useful—feature of categorical knowledge representation that is not easily replicated in a purely logical system.

7. Types and the open-world assumption

The use of *types* is another distinctive feature of categorical knowledge representation. Unlike instance data, types can be added to existing logical systems without too much difficulty; we shall do so in Section 8. Nonetheless, we regard types as distinctive because category theory is typed "by default," while logic is not, and because, as a practical matter, the knowledge representation systems in common use are untyped. In this section, we discuss the significance of types for knowledge representation.

The reader may be puzzled by the claim that knowledge representation frameworks based on description logic are untyped. Isn't the assignment of individuals to concepts a form of typing? Indeed, isn't a primary purpose of description logic to taxonomize the types of things existing in a given domain via a hierarchy of interrelated concepts? If so, what could be the purpose of adding a *second*, explicit form of typing to the system? These questions have merit and we shall try to answer them in this section.

First, we explain why description logic is untyped. Consider the relation "friend of" from the FOAF ontology (Section 5.1). We want to express that only people can be friends. In description logic, we would use a value restriction (see Sections 2 and 3) to ensure that any two individuals in the "friend of" relation belong to the concept "Person." Then, given any two individuals belonging to the disjoint concept "Organization," the answer

to the question "Are the two entities friends?" would be "No." On one view that is a perfectly reasonable answer. But on another it is confused. We might argue that the answer to the question is neither "yes" nor "no" because the question does not make sense. Organizations are simply not the kind of things that can be friends with each other. Merely by asking whether two organizations are friends, we commit a *category mistake*—using "category" in the sense of Gilbert Ryle, not Eilenberg and Mac Lane. A programmer would call it a *type error*. These two possible responses illustrate the philosophical difference between classes and types.

More prosaically, in description logic any concepts are comparable, while in relational ologs only relations with the same domain and codomain are comparable. Thus, in description logic, there is a *universal concept*, to which all individuals belong, and it is possible to take the intersection of any two concepts. By contrast, a relational olog has only *local* maxima and *local* intersections within each collection of typed relations $X \to Y$.

These distinctions have practical implications for knowledge representation. To bring this out, we consider two different methods of constructing a *taxonomy* of entities in a relational olog. This task, although probably overrepresented in KR research, is important in many applications. The first method, effectively untyped, is based on subsumption of concepts with a single domain type. The second method creates a hierarchy of different types connected by inclusion maps.

We shall see that the difference between the two methods is related to the open and closed world assumptions in database theory and knowledge representation [Rei78]. Under the *open world assumption*, any statements that are not deducible from a knowledge base are not assumed to be either true or false. Under the *closed world assumption*, certain statements that are not deducible are assumed to be false. The open world assumption is the standard mode of reasoning in logical systems, including first-order logic and description logic. The closed world assumption is commonly used in databases and ruled-based knowledge representation. For example, in the database for the FOAF ontology (Section 6.1), the absence of a row in the "friend of" table for Alice and Carol is interpreted as the absence of friendship between Alice and Carol—not the absence of *knowledge* of whether Alice and Carol are friends. More generally, closed world reasoning makes assumptions about what *is* true based on what is *not* explicitly stated.

To create a taxonomy in the style of description logic, we work with concepts $C : X \to I$ over a fixed domain type X. The subsumption axiom $C \Rightarrow D$ asserts that every instance of concept $C : X \to I$ is also an instance of concept $D : X \to I$. A collection of such axioms implicitly defines a hierarchy of concepts and sub-concepts. Inferences about concepts are made under the open world assumption. In particular, no two concepts C, D are provably disjoint unless there is a disjointness axiom $C \cap D \Rightarrow \perp$ (or disjointness can be inferred from other axioms). Relational ologs support disjointness axioms through the extensions of Section 9.

In a relational olog, it is also possible to represent a taxonomy as a hierarchy of types. For each kind in the taxonomy we define a type X. To declare that type X is a subtype of type Y, we add an inclusion map $\iota : X \to Y$. An *inclusion* of X into Y is simply a

morphism $\iota : X \to Y$ that is an injective map. Injectivity is asserted by the axiom
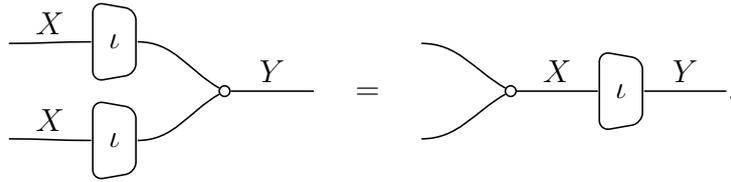
Given instance data $F : \mathcal{B} \to \mathbf{Rel}$, the function $F(\iota)$ associates each element of $F(X)$ with a unique element of $F(Y)$, thereby identifying $F(X)$ with a subset of $F(Y)$. In this way we interpret X as a subtype of Y. Inferences about types in the hierarchy are made under what amounts to a closed world assumption: unless explicitly stated, distinct types are unrelated. Two types unconnected by any morphism are not merely "disjoint," they occupy different universes. It is not permitted to contemplate their intersection.

These design patterns are not mutually exclusive, and we expect that they can be profitably combined. On one hand, the open world assumption enables inference about concepts that are not recorded by, or even anticipated by, the creator of the ontology. It embodies the ethos of the Semantic Web, which, like the World Wide Web, allows "anyone to say anything about anything." On the other hand, it can be inconvenient in scientific domains with a large and tightly controlled vocabulary, such as biology and biomedicine. The Gene Ontology, for example, contains tens of thousands of concepts related to genes and their biological functions [Ash+00]. Many of them are disjoint, and all these constraints must be recorded. Graphical ontology editors like Protégé simplify these tasks [Mus15], but omissions are still easy to make. We suggest that a judicious use of typing could eliminate the most embarrassing errors of this kind.

Some authors have tried to augment description logics like OWL with closed world reasoning [KAH11; SKH11]. This work is partly motivated by the need for closed world reasoning when an ontology is used "like a database" to make inferences about particular individuals. In this context, it is instance data, not typing, that offers a simple solution. Given a relational olog $\mathcal{B}$ and instance data $D : \mathcal{B} \to \mathbf{Rel}$, inference in $\mathcal{B}$ is open world (modulo constraints imposed by the type system). By contrast, inference in the bicategory of relations $D(\mathcal{B})$, a subcategory of $\mathbf{Rel}$, is closed world. For any two relations $R, S : X \to Y$, there is a subsumption $D(R) \Rightarrow D(S)$ in the database if and only if the table $D(R)$ is a subset of the table $D(S)$. Assuming the database has complete information with respect to a given population, if a subsumption $D(R) \Rightarrow D(S)$ holds at the instance level, then we can regard the subsumption $R \Rightarrow S$ as valid for that population, even if $R \Rightarrow S$ cannot be deduced at the knowledge level (in $\mathcal{B}$). This is a form of closed world reasoning. Again, we see the utility of a clean separation between universal and particular knowledge.

Another important use of types is to represent "concrete data types" like integers, real numbers, and strings within an ontology. All practical description logic systems, including OWL, support data types, but only though ad hoc extensions of the logical language [Baa+07, §6.2]. We grant that software implementations may need to handle primitive data types specially, but think it inelegant to distinguish in the mathematical formalism between "abstract" types like "Person" and "Organization" and "concrete" types like numbers and strings.

As others have observed, category theory builds a bridge between traditional mathematical logic and programming language theory. In this setting, it connects description logic with

type theory. Relational ologs are based on simple type theory. Of the two basic algebraic data types, we already have *product types*, and we shall introduce *sum types* in Section 9. We could conceivably use a more sophisticated type system. For example, instead representing subtypes by inclusions, we could add first-class subtypes and polymorphism. Polymorphic and other type theories have been extensively investigated in the context of categorical logic [Cro93; Jac99].

In the programming language community, it is generally accepted that some amount of typing increases the robustness and maintainability of software systems (although opinions differ greatly as to how much typing is desirable). Apart from low-level assembly languages, there are virtually no programming languages in common use that are completely untyped. In the same spirit, we argue that at least some typing is desirable in knowledge representation systems. The extent and sophistication of the typing will depend on the application and on personal preferences.

8. Categorical logic

There is a fundamental connection between relational ologs and logical formalisms for knowledge representation. We foreshadowed this connection in Section 3 by defining the structures that make **Rel** into a bicategory of relations using logical, rather than set-theoretic, notation. In fact, it is possible to reason about *any* bicategory of relations using first-order logic, in effect pretending that it is **Rel**. This conclusion is perhaps surprising, since some bicategories of relations, such as the category **VectRel** of linear relations (Section 6.4), look "from the outside" quite different than **Rel**.

The purpose of this section, and the attendant Appendix A, is to make precise the connection between relational ologs and first-order logic. Our results belong to *categorical logic*, which represents both syntax and semantics as categories and interpretations of logical theories as functors. The field was initiated by Lawvere's seminal thesis on the functorial semantics of algebraic theories [Law63]. An important result of categorical logic, perhaps the best known application of category theory to computer science, is that the simply-typed lambda calculus is the *internal language* of cartesian closed categories [LS88; Cro93; Jac99]. In a similar spirit, we prove that a certain fragment of first-order logic, called "regular logic," is the internal language of bicategories of relations.

We now explain semiformally the correspondence between regular logic and bicategories of relations. For details and proofs, we refer to Appendix A.

Regular logic is the fragment of first-order logic with connectives $\exists, \wedge, \top, =$. Unlike traditional first-order logic, regular logic is *typed* (cf. Section 7). Every variable x, free or bound, is assigned a type A, expressed by writing $x : A$. To indicate the types of free variables, every formula of regular logic is associated with a list of type assignments, called a *context*. Here are some examples of formulas in context:

$$x : A, y : B \mid R(x, y) \wedge S(x, y)$$
$$x : A, z : C \mid \exists y : B.(R(x, y) \wedge S(y, z))$$
$$x : A, x' : A, x'' : A \mid (x = x') \wedge (x = x'')$$
$$x : A \mid \top$$

36

In general, a formula in context has form $\Gamma \mid \varphi$, where φ is a formula and Γ is a context containing all the free variables of φ.

A theory in regular logic, or *regular theory*, is defined by the following data. There is a set of basic types $A, B, \ldots$ and a set of relation symbols $R, S, \ldots$. Each relation symbol R has a fixed signature $(A_1, \ldots, A_n)$ that determines its arity and argument types. A regular theory also has a set of *axioms* of form $\Gamma \mid \varphi \vdash \psi$, which we interpret as: "by assumption, φ implies ψ in the context Γ." Given a regular theory $\mathbb{T}$, we say that $\Gamma \mid \varphi$ *entails* $\Gamma \mid \psi$ under the theory $\mathbb{T}$, or that $\Gamma \mid \varphi \vdash \psi$ is a *theorem* of $\mathbb{T}$, if $\Gamma \mid \psi$ can be deduced from $\Gamma \mid \varphi$ using the axioms of $\mathbb{T}$ and the inference rules of the proof system for regular logic.

As an example, we define a regular theory capturing a fragment of the "friend of a friend" ontology (Section 5.1). The types of the theory are "Person," "Organization," "Number," and "String." Its relation symbols include

$$\text{knows} : (\text{Person}, \text{Person}), \qquad \text{friend of} : (\text{Person}, \text{Person}),$$
$$\text{works at} : (\text{Person}, \text{Organization}), \qquad \text{salary} : (\text{Person}, \text{Organization}, \text{Number}).$$

The symmetry of the "knows" relation is expressed by the axiom

$$x : \text{Person}, y : \text{Person} \mid \text{knows}(x, y) \vdash \text{knows}(y, x).$$

The "works at" relation is determined by

$$x : \text{Person}, y : \text{Organization} \mid \exists z : \text{Number}.(\text{salary}(x, y, z)) \dashv\vdash \text{works at}(x, y),$$

where, as usual, $\dashv\vdash$ is shorthand for $\vdash$ and $\dashv$ (two axioms).

We establish a correspondence between regular theories and bicategories of relations. First, to every regular theory $\mathbb{T}$, we associate a bicategory of relations $\text{Cl}(\mathbb{T})$, called the *classifying category* of $\mathbb{T}$. The classifying category is constructed directly from the syntax of regular logic. Its objects are finite lists of types $\underline{A} = (A_1, \ldots, A_n)$, which can be regarded as α-equivalence classes $[\Gamma]$ of contexts $\Gamma = \underline{x} : \underline{A}$. Its morphisms are equivalence classes of formulas in context $[\Gamma; \Delta \mid \varphi] : [\Gamma] \to [\Delta]$, where equality of formulas is up to α-equivalence (renaming of variables) and deducible logical equivalence under the axioms of $\mathbb{T}$. For instance, in the "friend of a friend" theory, we have

$$[x : \text{Person}; y : \text{Org} \mid \exists z : \text{Number}.(\text{salary}(x, y, z))] = [a : \text{Person}; b : \text{Org} \mid \text{works at}(a, b)].$$

The semicolon in the context partitions the free variables into domain and codomain; it serves no logical purpose. To make $\text{Cl}(\mathbb{T})$ into a bicategory of relations, we define composition, products, and diagonals analogously to **Rel** (Section 3).

Conversely, to every small bicategory of relations $\mathcal{B}$, we associate a regular theory $\text{Lang}(\mathcal{B})$, called the *internal language* of $\mathcal{B}$. Its types are the objects of $\mathcal{B}$ and its relation symbols are the morphisms of $\mathcal{B}$. Note that $\text{Lang}(\mathcal{B})$ necessarily has infinitely many types and relation symbols! The axioms of $\text{Lang}(\mathcal{B})$ are chosen to guarantee that formulas corresponding to equal morphisms are provably equivalent. To make sense of this statement, we must explain how an arbitrary formula in context $\Gamma \mid \varphi$ of $\text{Lang}(\mathcal{B})$ can be *interpreted* as a morphism $[\![\Gamma \mid \varphi]\!]$ of $\mathcal{B}$. The mapping $[\![\cdot]\!]$ is essentially inverse to the constructions making the classifying category into a bicategory of relations. We prove a *soundness theorem* for

general interpretations of a regular theory $\mathbb{T}$ in a bicategory of relations $\mathcal{B}$, yielding a *categorical semantics* for regular logic.

The main result of this section, proved in Appendix A, is that Cl is inverse to Lang in an appropriate sense.

Theorem. *With respect to typed regular logic, for every small bicategory of relations $\mathcal{B}$, there is an equivalence of categories*

$$\mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \simeq \mathcal{B} \qquad in \qquad \mathbf{BiRel}.$$

Consequently, we can regard regular theories and small bicategories of relations as "the same." Besides enriching our understanding of relational ologs, this result is potentially practically useful, as it enables the transfer of tools and techniques between category theory and logic. What should it mean to give "instance data" for a description logic knowledge base, assuming it can be expressed as a regular theory $\mathbb{T}$? We simply ask what data is required to give a structure-preserving functor $D : \mathrm{Cl}(\mathbb{T}) \to \mathbf{Rel}$. What should be meant by a "translation" $\mathbb{T} \to \mathbb{T}'$ between knowledge bases $\mathbb{T}$ and $\mathbb{T}'$? Again, we need only ask what data is needed to give a functor $F : \mathrm{Cl}(\mathbb{T}) \to \mathrm{Cl}(\mathbb{T}')$. We will not carry out these exercises here but it is instructive to do so.

In the other direction, we can reason about a relational olog $\mathcal{B}$ by performing logical inference in $\mathrm{Lang}(\mathcal{B})$. This observation is significant because the computational aspects of category theory are highly underdeveloped in comparison with logic. Much research on description logic is directed towards its computability and complexity theory, and there is a long tradition of computational first-order logic. Many mature inference engines and theorem provers exist. By contrast, the theory and practice of computational category theory, especially higher category theory, is only now emerging [Mim14; KZ15; BKV16].

Bibliographic remarks   Regular logic has been thoroughly studied by categorical logicians as the simplest fragment of first-order logic with a quantifier [Awo09; But98; Oos95]. Our categorical semantics of regular logic is quite different from the usual one. Conventionally, a formula $\Gamma \mid \varphi$ is interpreted as a *subobject* $\llbracket \Gamma \mid \varphi \rrbracket$ of the object $\llbracket \Gamma \rrbracket$: an equivalence class of monomorphisms into $\llbracket \Gamma \rrbracket$. A category suitable for such interpretations, called a *regular category*, has all finite limits and "well-behaved" subobjects. The classifying category of a regular theory is a regular category whose objects are (equivalence classes of) formulas in context and whose morphisms are (equivalence classes of) formulas in context that are provably functional. In our framework, the classifying bicategory of relations has as objects (equivalence classes of) contexts and as morphisms (equivalence classes of) formulas in context. The latter perspective seems more natural to us.

The germ of the above theorem is present already in the original paper of Carboni and Walters [CW87, Remark 2.9 (iii)], but to our knowledge has never been carefully developed. There are also strong connections between bicategories of relations and regular categories. Given a regular category $\mathcal{C}$, there is a bicategory of relations $\mathrm{Rel}(\mathcal{C})$ with the same objects as $\mathcal{C}$ and with morphisms $A \to B$ equal to the subobjects of $A \times B$—a construction that predates and motivates Carboni and Walters' paper. Yet not every bicategory of relations arises in this way. Conversely, if a bicategory of relations $\mathcal{B}$ is functionally complete, then $\mathrm{Map}(\mathcal{B})$ is a regular category [CW87, Theorem 3.5]. Yet in general $\mathrm{Map}(\mathcal{B})$ need not be

a regular category. Thus, to a limited extent, it is possible to pass between bicategories of relations and regular categories.

9. More expressive relational ologs

Relational ologs, as developed so far, can express local intersections and maxima. As proved in Section 8, their internal language is the regular $(\exists, \wedge, \top, =)$ fragment of first-order logic. It is natural to ask for more expressive relational ologs allowing local unions and minima. In logical terms, they should correspond to the coherent $(\exists, \wedge, \vee, \top, \bot, =)$ fragment of first-order logic. In this section, we develop such highly expressive relational ologs, called *distributive relational ologs*. We follow the pattern established by Sections 3 to 5: first, we present the relevant monoidal structures on **Rel**; next, we abstract from **Rel** to formulate a general categorical structure, called a *distributive bicategory of relations*; finally, we define a distributive relational olog to be a finitely presented distributive bicategory of relations.

9.1. The category of relations, revisited

The category of relations has another interesting monoidal product, besides the Cartesian product: the disjoint union. In this section, we explain disjoint unions from a categorical perspective. As in Section 3, our presentation draws on the survey [CP10], especially §3.5 on "classical-like" monoidal products.

The *disjoint union* (or *tagged union*) is defined on objects of **Rel** by

$$X \oplus Y := \{(x, 1) : x \in X\} \cup \{(y, 2) : y \in Y\}.$$

An element of $X \oplus Y$ is either an element of X or an element of Y, plus a special tag to avoid ambiguity when X and Y intersect. Given morphisms $R : X \to Y$ and $S : Z \to W$ of **Rel**, the *disjoint union* $R \oplus S : X \oplus Z \to Y \oplus W$ is defined by

$$(t, i)(R \oplus S)(s, j) \qquad \text{iff} \qquad \begin{cases} R(t, s) & \text{if } i = j = 1 \\ S(t, s) & \text{if } i = j = 2 \\ \bot & \text{otherwise} \end{cases}$$

The monoidal unit is $O := \emptyset$, the empty set. Finally, the braiding morphism $\boldsymbol{\sigma}_{X,Y} : X \oplus Y \to Y \oplus X$ exchanges the tags. With these definitions, $(\mathbf{Rel}, \oplus, O)$ is a symmetric monoidal category.

The category of relations is now equipped with two monoidal products. In general, when working with two monoidal products $\otimes$ and $\oplus$, we call the first product $\otimes$ the *tensor* and the second product $\oplus$ the *cotensor*. To avoid confusion, we will always use "light" notation for structures associated with the tensor and "dark" notation for structures associated with the cotensor. What that means should become clear shortly.

We would like to reason about both monoidal products using a single graphical language. Unfortunately, that is not entirely straightforward. The basic problem is that we now have

an effectively *three*-dimensional language, with dimensions corresponding to composition, the tensor, and the cotensor, but drawing pictures in dimensions greater than two is highly impractical. We discuss (two-dimensional!) graphical languages for multiple products below. For the moment, we work exclusively with the cotensor and can therefore employ, without ambiguity, the usual graphical language of monoidal categories.

We now consider structures derived from the disjoint union. Define the codiagonals $\blacktriangledown_X : X \oplus X \to X$ and $\blacksquare_X : O \to X$ on **Rel** by

$$(x, i)\blacktriangledown_X x' \qquad \text{iff} \qquad x = x'.$$

Note that the initial morphism $\blacksquare_X$ must be the (typed) empty relation $O \to X$. Define the diagonals $\blacktriangle_X := \blacktriangledown_X^\dagger : X \to X \oplus X$ and $\blacklozenge_X := \blacksquare_X^\dagger : X \to O$ by duality. As in $(\textbf{Rel}, \otimes, I)$, these morphisms form special $\dagger$-Frobenius monoids; in particular, they satisfy the Frobenius equations.

The disjoint union is logically dual to the Cartesian product. The union $R \cup S$ of two relations $R, S : X \to Y$ is $\blacktriangle_X(R \oplus S)\blacktriangledown_Y$ or, in graphical language,

$$R \cup S \quad = \quad$$
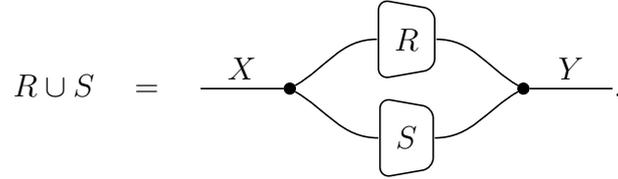
Similarly, the typed empty relation $\perp_{X,Y} : X \to Y$, or local minimum, is $\blacklozenge_X \cdot \blacksquare_Y$:

$$\perp_{X,Y} \quad = \quad$$

In particular, the boolean $\perp : I \to I$ is $\perp_{I,I} = \blacklozenge_I \cdot \blacksquare_I$.

There is a categorical interpretation of logical duality. The familiar principle of 1-categorical duality establishes a correspondence between a category $\mathcal{C}$ and its opposite $\mathcal{C}^{\text{op}}$. In a bicategory of relations, this form of duality is captured by the dagger functor. By analogy, in a 2-category $\mathcal{B}$, we can consider the 2-category $\mathcal{B}^{\text{co}}$ obtained from $\mathcal{B}$ by reversing all 2-morphisms. The correspondence between $\mathcal{B}$ and $\mathcal{B}^{\text{co}}$ is duality at the level of 2-morphisms. If $\mathcal{B}$ is a bicategory of relations, then 2-categorical duality is logical duality. Consider the situation in **Rel**. The diagonals Δ_X and $\lozenge_X$ are maps, and the codiagonals $\blacktriangledown_X$ and $\blacksquare_X$ are also maps. Equivalently, the diagonals Δ_X and $\lozenge_X$ are maps in **Rel**, while the diagonals $\blacktriangle_X$ and $\blacklozenge_X$ are maps in $\textbf{Rel}^{\text{co}}$. Thus $(\textbf{Rel}^{\text{co}}, \oplus, O)$ is *also* a bicategory of relations, provided we verify the axiom on lax monoid homomorphisms.

In fact, a stronger statement holds, breaking the symmetry between products and sums. Unlike the diagonals Δ_X and $\lozenge_X$ in $(\textbf{Rel}, \otimes, I)$, the codiagonals $\blacktriangledown_X$ and $\blacksquare_X$ in $(\textbf{Rel}, \oplus, O)$ are *natural*. That is, for every relation $R : X \to Y$, we have $(R \oplus R)\blacktriangledown_Y = \blacktriangledown_X R$ and $\blacksquare_X R = \blacksquare_Y$, or graphically
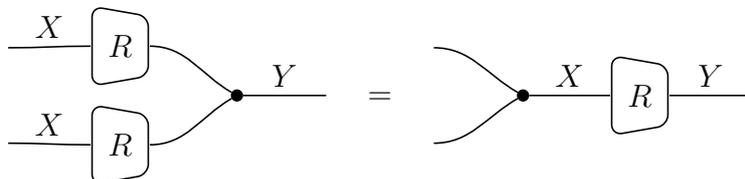
and

$$\bullet \!\!\!\! \xrightarrow{\ X\ } \boxed{R} \xrightarrow{\ Y\ } \quad = \quad \bullet \!\!\!\! \xrightarrow{\ Y\ }.$$

Likewise, we have $\blacktriangle_X(R \oplus R) = R\blacktriangle_Y$ and $R\blacklozenge_Y = \blacklozenge_X$. This situation motivates the following definition.

Definition. Let $(\mathcal{C}, \oplus, O)$ be a symmetric monoidal category. The monoidal product $\oplus$ on $\mathcal{C}$ is a

- a *(categorical) product* if there exists a diagonal $(\blacktriangle_A, \blacklozenge_A)$, natural in A;
- a *(categorical) coproduct* if there exists a codiagonal $(\blacktriangledown_A, \blacksquare_A)$, natural in A;
- a *biproduct* if it both a product and coproduct, such that for any objects A_1, A_2, the projection maps $\pi_1 = 1_{A_1} \oplus \blacklozenge_{A_2}$ and $\pi_2 = \blacklozenge_{A_1} \oplus 1_{A_2}$, the inclusion maps $\iota_1 = 1_{A_1} \oplus \blacksquare_{A_2}$ and $\iota_2 = \blacksquare_{A_1} \oplus 1_{A_2}$, and zero maps $0_{i,j} = \blacklozenge_{A_i}\blacksquare_{A_j}$ satisfy the equations

$$\iota_i \cdot \pi_j = \delta_{i,j} := \begin{cases} 1_{A_i} & \text{if } i = j \\ 0_{i,j} & \text{if } i \neq j \end{cases}, \qquad i, j = 1, 2.$$

Remark. Although it is not immediately obvious, these definitions of "product" and "coproduct" agree with the standard definitions via universal properties [HV12].

The category of relations is a biproduct category with respect to the disjoint union. Another prime example of a biproduct category is $(\mathbf{Vect}_k, \oplus)$, the category of finite-dimensional vector spaces and linear maps, equipped with the direct sum. The category of linear relations, $\mathbf{VectRel}_k$, is *not* a biproduct category.

Interactions between monoidal products We have hitherto studied the disjoint union only in isolation. We now consider how the Cartesian product and disjoint union interact in $\mathbf{Rel}$. On the objects of $\mathbf{Rel}$, there is a natural isomorphism

$$X \otimes (Y \oplus Z) \cong (X \otimes Y) \oplus (X \otimes Z),$$

given by $(x, (w, i)) \mapsto ((x, w), i)$, that expresses the distributivity of products over sums. In words: having an element of X and an element of Y or Z is the same as having elements of X and Y or elements of X and Z. Here is one possible general definition of distributivity in a monoidal category.

Definition ([Jay93]). A *distributive monoidal category* is a symmetric monoidal category $(\mathcal{C}, \otimes, I)$ with coproduct $\oplus$ that satisfies the *distributive law*: for any objects A, B, C, the canonical distributivity morphism

$$(A \otimes B) \oplus (A \otimes C) \to A \otimes (B \oplus C),$$

determined by the universal property of the coproduct, is an isomorphism.

Remark. A *rig category*, or *bimonoidal category*, retains the distributive law but relaxes the requirements that the tensor be symmetric and that the cotensor be the coproduct. It categorifies the classical algebraic structure known as a "rig" (ring without negatives) [BD98]. For the purposes of this paper, the extra generality of rig categories is unnecessary.

Besides **Set** and **Rel**, examples of distributive monoidal categories include **Ab**, the category of abelian groups, and **Vect**$_k$, the category of finite-dimensional vector spaces, both equipped the tensor product $\otimes$ and the direct sum $\oplus$.

The distributive law extends to morphisms of **Rel** in a familiar way. For any three relations $R, S, T : X \to Y$, we have

$$R \cap (S \cup T) = (R \cap S) \cup (R \cap T).$$

It is tempting to display this equation diagrammatically:
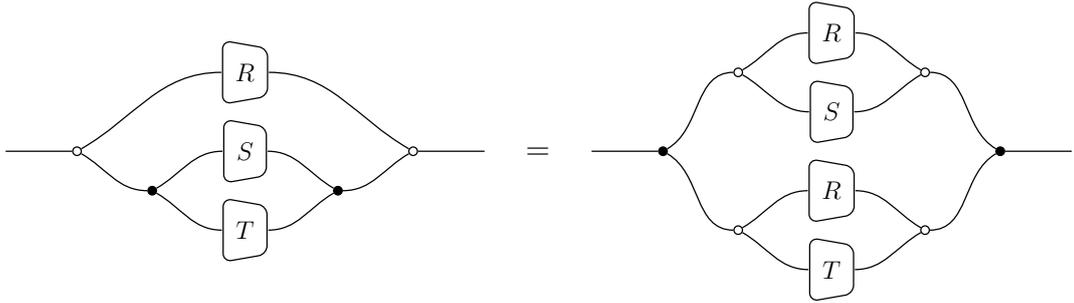
As noted at the beginning of this section, diagrams involving two monoidal products take us beyond the firmly established graphical language of monoidal categories. The picture above relies on *context* derived from the copy and merge nodes to determine which monoidal product is "active" at a given point. In this case the notation is unambiguous, but in general one must take care to avoid coherence problems, especially when working with the monoidal units. We conjecture that soundness is maintained if the cotensor is restricted to forming unions (via the morphisms $\blacktriangle_X$ and $\blacktriangledown_X$) and local minima (via $\blacklozenge_X$ and $\blacksquare_X$). This restricted language is already sufficient for applications that do not directly utilize sum types, a special case of some practical interest.

In the literature, *proof nets* are established as a graphical calculus for categories with two monoidal products. Girad introduced proof nets in his seminal paper on linear logic [Gir87]. Blute et al generalized the formalism to weakly distributive and $*$-autonomous categories (models of linear logic), adopting a graphical style reminiscent of string diagrams [Blu+96]. Unfortunately, proof nets are considerably more complicated than string diagrams, accommodating monoidal units through special "thinning links." In our view it remains an open problem to define a graphical language for categories with multiple monoidal products that is provably coherent—sound and complete—but still simple enough for practical use by nonspecialists.

9.2. Distributive bicategories of relations

Motivated by distributivity in the category of relations, we define a categorical abstraction called a "distributive bicategory of relations." We begin with the following more general definition.

Definition. A *union bicategory of relations* is a locally posetal 2-category $\mathcal{B}$, a symmetric monoidal category $(\mathcal{B}, \otimes, I)$ with diagonals $(X, \Delta_X, \Diamond_X)$, and a symmetric monoidal category $(\mathcal{B}, \oplus, O)$ with codiagonals $(X, \blacktriangledown_X, \blacktriangle_X)$, such that

- every morphism $R : X \to Y$ is a lax comonoid homomorphism and a lax monoid homomorphism;
- the morphisms $\Delta_X, \Diamond_X, \blacktriangledown_X, \blacksquare_X$ have right adjoints, denoted $\nabla_X, \square_X, \blacktriangle_X, \blacklozenge_X$;
- both pairs (Δ_X, ∇_X) and $(\blacktriangle_X, \blacktriangledown_X)$ obey the Frobenius equations.

Equivalently, a *union bicategory of relations* is a locally posetal 2-category $\mathcal{B}$ such that both $\mathcal{B}$ and $\mathcal{B}^{\mathrm{co}}$ are bicategories of relations (not necessarily with respect to the same monoidal product).

Remark. To our knowledge, this definition does not appear in the literature. Note that Johnstone's "union allegory" insists that unions are preserved by composition [Joh02, §A3.2]; under our definition of "union bicategory of relations," the strongest statement that can be made about the interaction between unions and composition is the logical dual of the modular law, $(R \cup TS^\dagger)S \subseteq RS \cup T$.

The definition postulates no relationship whatsoever between the two monoidal products. An abelian bicategory of relations (Section 6.4) is a union bicategory of relations where the two products coincide. In our main example **VectRel**$_k$, the union $L \cup M$ of two linear relations $L, M \subseteq U \oplus V$ is the vector space sum $L + M$. The other important special case of a union bicategory of relations is the distributive bicategory of relations.

Definition. A *distributive bicategory of relations* is a union bicategory of relations where the cotensor is the categorical coproduct.

We denote by **DistBiRel** the category of (small) distributive bicategories of relations and structure-preserving functors.

Remark. Carboni and Walters mention "distributive" bicategories of relations in passing, but do not clearly state a definition [CW87, Remark 3.7]. It seems likely that our definition is what they had in mind. Freyd and Scedrov utilize an analogous concept of "distributive allegory" [FS90].

Of course, **Rel** is a distributive bicategory of relations. Another example is the category of boolean matrices, **Mat**($\mathbb{B}$), introduced in Section 6.3.

Several important properties are implicit in the definition. As the name suggests, the distributive law holds automatically in a distributive bicategry of relations. In fact, if $(\mathcal{C}, \otimes, I)$ is any compact closed category and $\oplus$ is the coproduct, then $\mathcal{C}$ is a distributive monoidal category [Jay93]. Moreover, it can be shown that intersections distribute over unions in a distributive bicategory of relations. Also, the coproduct in a distributive bicategry of relations is automatically a biproduct, by the symmetry of a dagger category. (Alternatively, products or coproducts in a compact closed category are always biproducts [Hou08].) Table 3 summarizes the extra notation associated with a distributive bicategory of relations, extending Table 2 in Section 4.

Lastly, following the established pattern, we define a corresponding notion of olog. All remarks made in Section 5 about the meaning of "finitely presented" remain in force.

Definition. A *distributive relational olog* is a finitely presented distributive bicategory of relations.

Structure	Name	Notation and definition
biproduct category	biproduct	$R \oplus S$
	braiding	$\boldsymbol{\sigma}_{X,Y}$
	merge	$\blacktriangledown_X$
	create	$\blacksquare_X$
	copy	$\blacktriangle_X := \blacktriangledown_X^* = \blacktriangledown_X^\dagger$
	delete	$\blacklozenge_X := \blacksquare_X^* = \blacksquare_X^\dagger$
logical	union	$R \cup S := \blacktriangle_X (R \oplus S) \blacktriangledown_Y$
	false	$\bot := \blacklozenge_I \cdot \blacksquare_I$
	local minimum	$\bot_{X,Y} := \blacklozenge_X \cdot \blacksquare_Y$

Table 3: Summary of extra morphisms in a distributive bicategory of relations

Distributive relational ologs are very expressive. The only connectives of first-order logic not directly expressible are negation and universal quantification. However, the *negation*, or *complement*, of a relation $R : X \to Y$ can be implicitly defined by introducing another relation $S : X \to Y$ together with the two axioms $R \cap S \Rightarrow \bot_{X,Y}$ and $\top_{X,Y} \Rightarrow R \cup S$. In graphical language, the axioms are beautifully symmetric:
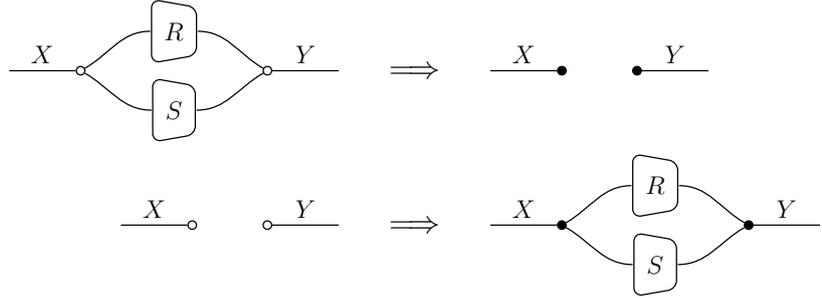
This definition of negation makes sense in any union bicategory of relations. In **Rel**, negation is the usual set-theoretic complement; in **VectRel**$_k$, it is the subspace complement (internal direct sum).

9.3. Categorical logic with product and sum types

By now it should be evident that distributive relational ologs correspond, in some sense, to the fragment of first-order logic with connectives $\exists, \wedge, \vee, \top, \bot, =$. This fragment is called *coherent logic* or, in older literature, *geometric logic*. Coherent logic, or variants thereof, has been used in axiomatic geometry [ADM09] and in automated theorem proving [Sto+14; GG16], in part because it is more readily interpretable by humans than richer logics. Coherent logic is nonetheless as expressive as first-order logic, in the sense that any first-order theory can be translated into an equivalent coherent theory called its *Morleyization* [Joh02, Lemma D1.5.13]. In the Morleyized theory, negations are encoded by the two axioms shown above.

Despite the suggestive analogy, a direct translation of Section 8 founders, due to the presence of sum types. In a bicategory of a relations, any two objects X and Y have a product $X \otimes Y$, but our system of typed regular logic does not include product types.

That is, the syntax does not permit the construction of a type $A \times B$ from two basic types A and B. This mismatch is, however, not fatal because products are smuggled into the logical system as *contexts*: a context $(x : A, y : B)$ amounts to a single variable of type $A \times B$. Adding a second monoidal product puts this device under considerable strain. One could conceivably extend the syntax of a context to represent a fully "destructured" element of arbitrary compound type. A better solution is to augment the logical language with product types $A \times B$ and sum types $A + B$, as well as a unit (singleton) type 1 and zero (empty) type 0. The role of contexts is downplayed accordingly.

Product and sum types are ubiquitous in programming language theory. The simply typed lambda calculus is often treated with product and sum types, in both classical and categorical settings [Sel13; LS88]. By contrast, first-order logics with non-trivial type systems are rare. A proof system for first-order logic with product and sum types does not, to our knowledge, appear in the literature. We now present such a system, straightforwardly adapted from the lambda calculus. As in Section 8, the treatment here is informal. Details and proofs are deferred to Appendix B.

Remark. A clarification may be helpful to readers acquainted with the Curry-Howard correspondence [How80; Wad15]. The interpretation of types as propositions, and programs as proofs, is the subject of a large body of research. However, we are interested in types *with* logic, not types *as* logic. The former is sometimes called "two-level type theory." An example is Gambino and Aczel's "logic-enriched type theory," a system of first-order logic with dependent types [GA06].

By using a non-trivial type system, we commit ourselves to a proper treatment of terms. In Section 8, we did not bother to distinguish the "terms" of regular logic, which are just typed variables $x : A$. We now formally distinguish two kinds of expressions in context: *formulas* (also called *propositions*) and *terms*. As before, the formulas are generated by the equality relation, relation symbols, and logical connectives. The terms are generated by variables, function symbols, and term constructors for the product and sum types.

The term constructors are familiar from typed lambda calculus. Given a term $t : A \times B$, there are *projection* terms $\pi_1(t) : A$ and $\pi_2(t) : B$, and given two terms $t : A$ and $s : B$, there is a *pair* term $\langle t, s \rangle : A \times B$. Dually, given terms $t : A$ and $s : B$, there are *inclusion* terms $\iota_1(t) : A + B$ and $\iota_2(s) : A + B$, and given terms $t : A + B$, $r : C$, $s : C$, there is a *copair* term $\delta(t, x : A.r, y : B.s) : C$. The copair term, or "case statement," is interpreted as follows. If $t : A + B$ is a value of type A, return the term r with variable x replaced by t. If $t : A + B$ is a value of type B, return the term s with variable y replaced by t. In either case, the result is a value of type C.

We now sketch the correspondence between coherent logic with product and sum types and distributive bicategories of relations. To every coherent theory $\mathbb{T}$ we associate the classifying category $\mathrm{Cl}(\mathbb{T})$, a distributive bicategory of relations. Its objects are the types of $\mathbb{T}$. Note the difference from Section 8, where the objects of $\mathrm{Cl}(\mathbb{T})$ are finite *lists* of types. The morphisms of $\mathrm{Cl}(\mathbb{T})$ are equivalence classes of formulas in context with exactly two free variables. The types of these variables are the domain and codomain of the morphism. We make the classifying category into a distributive bicategory of relations analogously to **Rel** (Sections 3 and 9.1).

Conversely, every small distributive bicategory of relations $\mathcal{B}$ has its internal language $\mathrm{Lang}(\mathcal{B})$, a coherent theory. As before, we interpret an arbitrary formula in context

$\Gamma \mid \varphi$ of Lang($\mathcal{B}$) as a morphism $[\![\Gamma \mid \varphi]\!]$ of $\mathcal{B}$. The construction proceeds analogously to Section 8, with one important addition: terms are interpreted as *maps*. More precisely, a term in context $\Gamma \mid t : A$ is interpreted as a morphism $[\![\Gamma \mid t : A]\!] : [\![\Gamma]\!] \to [\![A]\!]$ of Map($\mathcal{B}$). Although the connection between terms and maps is interesting its own right, in the present proof, the interpretation of terms as maps serves only to establish the base case in the inductive interpretation of formulas as morphisms.

Our main result, proved in Appendix B, is stated below.

Theorem. *With respect to coherent logic with product and sum types, for every small distributive bicategory of relations $\mathcal{B}$, there is an equivlance of categories*

$$\mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \simeq \mathcal{B} \qquad in \qquad \mathbf{DistBiRel}.$$

It is also possible to realize the correspondence between regular logic and bicategories of relations using a richer type system than in Section 8. The appropriate logic is regular logic with product types and a singleton type (but not sum types or an empty type).

Theorem. *With respect to regular logic with product types, for every small bicategory of relations $\mathcal{B}$, there is an equivalence of categories*

$$\mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \simeq \mathcal{B} \qquad in \qquad \mathbf{BiRel}.$$

10. Conclusion and outlook

In this paper, we have propounded a categorical framework for knowledge representation centered around bicategories of relations. We emphasized three important features that emerge automatically from category theory: instance data, types, and graphical syntax. We compared our framework informally to description logic and formally to the regular and coherent fragments of typed first-order logic. In this final section, we offer a general perspective on categorical knowledge representation. We also suggest directions for future research.

We have extensively discussed the relationship between the algebraic and logical approaches to knowledge representation, but have said comparatively little about how the two categorical frameworks—functional and relational ologs—are related. Although a complete answer is beyond the scope of this work, we will suggest a "pattern" or "template" for defining categorical ontologies that is general enough to encompass these and other frameworks. This template can perhaps serve as a first step towards a unified methodology of categorical knowledge representation.

Doctrines are a useful organizing principle for category theory [KR77]. Informally, a *doctrine* is a family of categories or higher categories with extra structure. The most basic doctrine is the doctrine of categories (with no extra structure). There are also doctrines of categories with finite products, of symmetric monoidal categories, of compact closed categories, of 2-categories, of bicategories of relations, etc. Besides the categories themselves, a doctrine specifies the relevant kind of "structure-preserving" functors, and the natural transformations between these. The concept of doctrine can be formalized, but for us this informal understanding is perfectly adequate.

Doctrine	Prototype(s)	Internal language
category	**Set**	
category with finite (co)limits	**Set**	
bicategory of relations	**Rel**	regular logic with product types
distributive bicategory of relations	**Rel**	coherent logic with product and sum types
cartesian closed category	**Set**, **CPO**	typed lambda calculus with product types
bicartesian closed category	**Set**	typed lambda calculus with product and sum types

Table 4: Selected doctrines relevant to knowledge representation

Here is a general recipe for constructing a categorical knowledge base. First, choose a doctrine. This choice should be informed by the phenomena being modeled; we expand on this idea below. Next, define a finitary specification language for the doctrine. As above, the basic strategy is finite presentation, a.k.a. the method of generators and relations, which works in considerable generality. If the doctrine supports arbitrary limits or colimits, the method of *sketches* can be used instead [Wel93; Mak97]. Alternatively, if the doctrine has as its internal language some well-known logical system, that system could serve as a specification language. Finally, use the specification language to define an ontology. As a mathematical object, the ontology is simply a finitely generated category of the doctrine. Instance data for the ontology is derived from the "prototype" category of the doctrine, such as **Set** or **Rel**, or possibly from a more exotic category.

Some doctrines relevant to knowledge representation are listed in Table 4. Functional ologs arise from the doctrine of categories with finite limits and colimits; relational ologs from the doctrine of bicategories of relations. The typed lambda calculus is the internal language of the doctrine of cartesian closed categories. Besides its central role in programming language theory, the lambda calculus has been used to model natural languages [HK98]. This list by no means exhausts the doctrines that are potentially useful for knowledge representation.

Different doctrines are appropriate for different applications. Bicategories of relations are designed to model classes of entities (concepts) and the relationships between them (roles). Description logic shares this orientation. However, creating taxonomies of concepts is hardly the only worthwhile application of knowledge representation. As an example, the original impetus for this work was our need to model knowledge about computer programs used in data analysis. Description logics (and relational ologs) are ill-suited to this project; a doctrine related to the lambda calculus is much more appropriate. In general, we worry that the mainstream of KR research has unjustifiably privileged taxonomies over other kinds of knowledge. Our philosophy is that category theory is a universal modeling language enabling a more expansive understanding of knowledge representation. In the future, we hope to see practical, flexible knowledge representation systems that allow doctrines to be rapidly assembled from the categorical toolbox to meet the needs of particular applications.

There are myriad directions for future research on categorical knowledge representation. We mention a few that are directly relevant to relational ologs.

A glaring omission in this work is any discussion of automated inference. By contrast, computationally tractable inference has been the prime directive of the description logic community. The first step for our project is to acknowledge that inference in a relational olog is undecidable. This is true even without the extensions of Section 9. (One can see this algebraically, by reduction from the word problems for monoids or groups, or logically, by reduction from the decision problem for regular theories.)

There are two possible responses to the problem of undecidability. We could follow the DL community in imposing language restrictions to achieve provable computational tractability. The extensive DL literature would doubtless be very helpful in carrying out this program. However, we worry that imposing ad hoc restrictions would do irredeemable violence to the formalism's elegance and expressivity. A second approach is to allow an unrestricted language and settle for approximate inference. We share with Doyle and Patil the opinion that this approach is undervalued by the description logic community [DP91]. In the statistics and machine learning communities, the need for approximate inference in complex models is an accepted fact of life. The contrast is especially stark because inference in first-order theories is *harder* than inference in probabilistic models. In any event, developing inference algorithms for relational ologs, exact or approximate, is an important prerequisite for practical applications.

Another problem, already raised in Section 9, is to define a graphical language for distributive bicategories of relations that is coherent, yet intuitive. (We think that proof nets fall short on the second count.) The graphical language of string diagrams is a very appealing feature of relational ologs. We hope that a satisfactory extension of string diagrams to categories with multiple monoidal products will be discovered.

Acknowledgments

References

[AC04]   Samson Abramsky and Bob Coecke. "A categorical semantics of quantum protocols". *Logic in Computer Science, 2004*. IEEE. 2004, pp. 415–425.

[ADM09] Jeremy Avigad, Edward Dead, and John Mumma. "A formal system for Euclid's Elements". *The Review of Symbolic Logic* 2.4 (2009), pp. 700–768.

[AG05]   Renzo Angles and Claudio Gutierrez. "Querying RDF data from a graph database perspective". *European Semantic Web Conference*. 2005, pp. 346–360.

[AG08]   Renzo Angles and Claudio Gutierrez. "Survey of graph database models".
            ACM Computing Surveys 40.1 (2008), p. 1.

[AL12]   Emilio Jesús Gallego Arias and James B. Lipton. "Logic programming in tabular allegories". *Technical Communications of the 28th International Conference on Logic Programming (ICLP'12)*. Vol. 17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 334–347.

[Apa15]   Apache Software Foundation. *TinkerPop3 Documentation*. 2015. URL: `http://tinkerpop.apache.org/docs/current/reference/` (visited on 03/23/2017).

[Ash+00]   Michael Ashburner et al. "Gene Ontology: tool for the unification of biology". *Nature Genetics* 25.1 (2000), pp. 25–29.

[Awo09]   Steve Awodey. "Introduction to Categorical Logic". Lecture notes. 2009. URL: `http://www.andrew.cmu.edu/user/awodey/catlog/notes/`.

[Awo10]   Steve Awodey. *Category Theory*. 2nd ed. Oxford University Press, 2010.

[Baa+07]   Franz Baader, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation and Applications*. 2nd ed. Cambridge University Press, 2007.

[BD98]   John Baez and James Dolan. "Categorification". *arXiv:math/9802029* (1998).

[BE15]   John Baez and Jason Erbele. "Categories in control". *Theory and Applications of Categories* 30.24 (2015), pp. 836–881.

[BF15]   John Baez and Brendan Fong. "A compositional framework for passive linear networks". *arXiv:1504.05625* (2015).

[BH94]   Carolyn Brown and Graham Hutton. "Categories, allegories and circuit design". *Logic in Computer Science, 1994. LICS'94*. IEEE. 1994, pp. 372–381.

[BHL01]   Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". *Scientific American* 284.5 (2001), pp. 28–37.

[BJ94]   Carolyn Brown and Alan Jeffrey. "Allegories of circuits". *International Symposium on Logical Foundations of Computer Science*. Springer. 1994, pp. 56–68.

[BKV16]   Krzysztof Bar, Aleks Kissinger, and Jamie Vicary. "Globular: an online proof assistant for higher-dimensional rewriting". *arXiv:1612.01093* (2016).

[BL04]   Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Elsevier, 2004.

[BL84]   Ronald Brachman and Hector Levesque. "The tractability of subsumption in frame-based description languages". *AAAI-84 Proceedings*. AAAI. 1984, pp. 34–37.

[Blu+96]   Richard F Blute, J Robin B Cockett, Robert A G Seely, and Todd H Trimble. "Natural deduction and coherence for weakly distributive categories". *Journal of Pure and Applied Algebra* 113.3 (1996), pp. 229–296.

[BM14]   Dan Brickley and Libby Miller. *FOAF Vocabulary Specification 0.99*. 2014. URL: `http://xmlns.com/foaf/spec/20140114.html`.

[BS10]   John Baez and Mike Stay. "Physics, topology, logic and computation: a Rosetta Stone". *New Structures for Physics*. Springer, 2010, pp. 95–172.

[BSZ14] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. "A categorical semantics of signal flow graphs". *International Conference on Concurrency Theory (CONCUR 2014)*. 2014, pp. 435–450.

[But98] Carsten Butz. "Regular categories and regular logic". *BRICS Lecture Series LS-98-2* (1998).

[Car+08] Aurelio Carboni, G Max Kelly, Robert FC Walters, and Richard J Wood. "Cartesian bicategories II". *Theory and Applications of Categories* 19.6 (2008), pp. 93–124.

[Car86] John Cartmell. "Generalised algebraic theories and contextual categories". *Annals of Pure and Applied Logic* 32 (1986), pp. 209–243.

[CLW93] Aurelio Carboni, Stephen Lack, and Robert FC Walters. "Introduction to extensive and distributive categories". *Journal of Pure and Applied Algebra* 84.2 (1993), pp. 145–158.

[Cod70] Edgar F Codd. "A relational model of data for large shared data banks". *Communications of the ACM* 13.6 (1970), pp. 377–387.

[CP10] Bob Coecke and Eric Oliver Paquette. "Categories for the practising physicist". *New Structures for Physics*. Springer, 2010, pp. 173–286.

[Cro93] Roy L Crole. *Categories for Types*. Cambridge University Press, 1993.

[CW87] Aurelio Carboni and Robert FC Walters. "Cartesian bicategories I". *Journal of Pure and Applied Algebra* 49.1-2 (1987), pp. 11–32.

[DP91] Jon Doyle and Ramesh S Patil. "Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services". *Artificial intelligence* 48.3 (1991), pp. 261–297.

[DV10] Ian Davis and Eric Vitiello Jr. *RELATIONSHIP: A vocabulary for describing relationships between people*. 2010. URL: http://vocab.org/relationship/.

[FDB06] Marcelo Fiore, Roberto Di Cosmo, and Vincent Balat. "Remarks on isomorphisms in typed lambda calculi with empty and sum types". *Annals of Pure and Applied Logic* 141.1-2 (2006), pp. 35–50.

[Fre72] Peter Freyd. "Aspects of topoi". *Bulletin of the Australian Mathematical Society* 7.1 (1972), pp. 1–76.

[Fri12] Bernard Friedland. *Control system design: an introduction to state-space methods*. Courier Dover Publications, 2012.

[FS90] Peter J Freyd and Andre Scedrov. *Categories, Allegories*. Elsevier, 1990.

[GA06] Nicola Gambino and Peter Aczel. "The generalised type-theoretic interpretation of constructive set theory". *Journal of Symbolic Logic* (2006), pp. 67–103.

[Gal+93] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. "Directed hypergraphs and applications". *Discrete Applied Mathematics* 42.2 (1993), pp. 177–201.

[GG16] M Ganesalingam and W T Gowers. "A fully automatic theorem prover with human-style output". *Journal of Automated Reasoning* (2016), pp. 1–39.

[Gir87]   Jean-Yves Girard. "Linear logic". *Theoretical Computer Science* 50.1 (1987), pp. 1–101.

[Gra+08]   Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. "OWL 2: The next step for OWL". *Web Semantics: Science, Services and Agents on the World Wide Web* 6.4 (2008), pp. 309–322.

[Har09]   John Harding. "A link between quantum logic and categorical quantum mechanics". *International Journal of Theoretical Physics* 48.3 (2009), pp. 769–802.

[HK98]   Irene Heim and Angelika Kratzer. *Semantics in generative grammar*. Blackwell, 1998.

[HKR09]   Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. CRC Press, 2009.

[HKS05]   Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The irresistible $\mathcal{SHIQ}$". *Proc. of OWL: Experiences and Directions*. 2005.

[HKS06]   Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The even more irresistible $\mathcal{SROIQ}$". *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*. AAAI, 2006, pp. 57–67.

[Hou08]   Robin Houston. "Finite products are biproducts in a compact closed category". *Journal of Pure and Applied Algebra* 212.2 (2008), pp. 394–400.

[How80]   William A Howard. "The formulae-as-types notion of construction". *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalism* 44 (1980), pp. 479–490.

[HV12]   Chris Heunen and Jamie Vicary. *Lectures on categorical quantum mechanics*. Computer Science Department, Oxford University. 2012.

[Jac95]   Bart Jacobs. "Parameters and parametrization in specification, using distributive categories". *Fundamenta informaticae* 24.3 (1995), pp. 209–250.

[Jac99]   Bart Jacobs. *Categorical logic and type theory*. Vol. 141. Studies in Logic and the Foundations of Mathematics. Elsevier, 1999.

[Jay93]   C Barry Jay. "Tail recursion through universal invariants". *Theoretical Computer Science* 115.1 (1993), pp. 151–189.

[Joh02]   Peter T Johnstone. *Sketches of an elephant: A topos theory compendium*. 2 vols. Oxford University Press, 2002.

[KAH11]   Matthias Knorr, José Júlio Alferes, and Pascal Hitzler. "Local closed world reasoning with description logics under the well-founded semantics". *Artificial Intelligence* 175.9-10 (2011), pp. 1528–1554.

[KN94]   Petrus Knijnenburg and Frank Nordemann. *Two Categories of Relations*. Tech. rep. 94-32. Leiden University, Department of Computer Science, 1994.

[Koc04]   Joachim Kock. *Frobenius algebras and 2-d topological quantum field theories*. Cambridge University Press, 2004.

[KR77]   Anders Kock and Gonzalo Reyes. "Doctrines in categorical logic". *Handbook of Mathematical Logic*. Ed. by Jon Barwise. North Holland, 1977.

[KSH12]   Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. "A description logic primer". *arXiv:1201.4089* (2012).

[KZ15]   Aleks Kissinger and Vladimir Zamdzhiev. "Quantomatic: A proof assistant for diagrammatic reasoning". *International Conference on Automated Deduction.* Springer. 2015, pp. 326–336.

[Lac10]   Stephen Lack. "A 2-categories companion". *Towards higher categories.* 2010, pp. 105–191.

[Law15]   Finn Lawler. "Fibrations of predicates and bicategories of relations". *arXiv:1502.08017* (2015).

[Law63]   F. William Lawvere. "Functorial Semantics of Algebraic Theories". PhD thesis. Columbia University, 1963.

[Lei14]   Tom Leinster. *Basic category theory.* Vol. 143. Cambridge University Press, 2014.

[LS09]   F William Lawvere and Stephen H Schanuel. *Conceptual mathematics: a first introduction to categories.* 2nd ed. Cambridge University Press, 2009.

[LS88]   Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic.* Cambridge University Press, 1988.

[Mac63]   Saunders Mac Lane. "Natural associativity and commutativity". *Rice University Studies* 49.4 (1963), pp. 28–46.

[Mac98]   Saunders Mac Lane. *Categories for the working mathematician.* 2nd ed. Vol. 5. Graduate Texts in Mathematics. Springer-Verlag, 1998.

[Mah16]   Michael Mahoney. "Lecture notes on spectral graph methods". *arXiv:1608.04845* (2016).

[Mak97]   Michael Makkai. "Generalized sketches as a framework for completeness theorems. Part I". *Journal of Pure and Applied Algebra* 115.1 (1997), pp. 49–79.

[Mim14]   Samuel Mimram. "Towards 3-dimensional rewriting theory". *arXiv:1403.4094* (2014).

[Mus15]   Mark A Musen. "The Protégé project: A look back and a look forward". *AI Matters* 1.4 (2015), pp. 4–12.

[Nic+16] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A review of relational machine learning for knowledge graphs". *Proceedings of the IEEE* 104.1 (2016), pp. 11–33.

[Nie12]   Michael Nielsen. *Reinventing Discovery: The New Era of Networked Science.* Princeton University Press, 2012.

[Noy+09] Natalya F Noy et al. "BioPortal: ontologies and integrated data resources at the click of a mouse". *Nucleic Acids Research* (2009).

[Oos95]   Jaap van Oosten. "Basic Category Theory". *BRICS Lecture Series LS-95-1* (1995).

[Pat17]   Evan Patterson. *epatters/Catlab: An experimental library for computational category theory.* Apr. 2017. DOI: 10.5281/zenodo.569966.

[Pit95]   Andrew Pitts. *Categorical logic.* Tech. rep. University of Cambridge, Computer Laboratory, 1995.

[PS08]   Eric Prud'hommeaux and Andy Seaborne, eds. *SPARQL Query Language for RDF.* World Wide Web Consortium (W3C). 2008. URL: `https://www.w3.org/TR/rdf-sparql-query/`.

[PS97]   Frank Piessens and Eric Steegmans. "Proving semantical equivalence of data specifications". *Journal of Pure and Applied Algebra* 116.1-3 (1997), pp. 291–322.

[RB11]   Peter N Robinson and Sebastian Bauer. *Introduction to bio-ontologies.* CRC Press, 2011.

[Rei78]   Raymond Reiter. "On closed world data bases". *Logic and data bases.* Springer, 1978, pp. 55–76.

[Rie16]   Emily Riehl. *Category theory in context.* Dover, 2016.

[RN10]   Marko A Rodriguez and Peter Neubauer. "Constructions from dots and lines". *Bulletin of the American Society for Information Science and Technology* 36.6 (2010), pp. 35–41.

[Sch+16]   Patrick Schultz, David I Spivak, Christina Vasilakopoulou, and Ryan Wisnesky. "Algebraic databases". *arXiv:1602.03501* (2016).

[Sel07]   Peter Selinger. "Dagger compact closed categories and completely positive maps". *Electronic Notes in Theoretical Computer Science* 170 (2007), pp. 139–163.

[Sel10]   Peter Selinger. "A survey of graphical languages for monoidal categories". *New Structures for Physics.* Springer, 2010, pp. 289–355.

[Sel13]   Peter Selinger. "Lecture notes on the lambda calculus". *arXiv:0804.3434* (2013).

[Sel99]   Peter Selinger. "Categorical structure of asynchrony". *Electronic Notes in Theoretical Computer Science* 20 (1999), pp. 158–181.

[SK12]   David I Spivak and Robert E Kent. "Ologs: a categorical framework for knowledge representation". *PLoS One* 7.1 (2012).

[SKH11]   Kunal Sengupta, Adila Alfa Krisnadhi, and Pascal Hitzler. "Local closed world semantics: Grounded circumscription for OWL". *International Semantic Web Conference.* Springer. 2011, pp. 617–632.

[Spi12]   David I Spivak. "Functorial data migration". *Information and Computation* 217 (2012), pp. 31–51.

[Spi14]   David I Spivak. *Category theory for the sciences.* MIT Press, 2014.

[Sto+14]   Sana Stojanović, Julien Narboux, Marc Bezem, and Predrag Janičić. "A vernacular for coherent logic". *International Conference on Intelligent Computer Mathematics.* Springer. 2014, pp. 388–403.

[SW15]   David I Spivak and Ryan Wisnesky. "Relational foundations for functorial data migration". *Proceedings of the 15th Symposium on Database Programming Languages.* ACM. 2015, pp. 21–28.

[TG87]   Alfred Tarski and Steven Givant. *A formalization of set theory without variables.* Vol. 41. American Mathematical Society, 1987.

[Wad15]   Philip Wadler. "Propositions as types". *Communications of the ACM* 58.12 (2015), pp. 75–84.

[Wel93]   Charles Wells. "Sketches: Outline with References". With 2009 addendum. 1993. URL: `http://www.cwru.edu/artsci/math/wells/pub/pdf/Sketch.pdf`.

[Woo75]   William A Woods. "What's in a link: Foundations for semantic networks". *Representation and Understanding: Studies in Cognitive Science* (1975), pp. 35–82.

[ZMS13]   Bartosz Zieliński, Paweł Maślanka, and Ścibor Sobieski. "Allegories for database modeling". *International Conference on Model and Data Engineering.* Springer. 2013, pp. 278–289.

A. Regular logic and bicategories of relations

The objective of this appendix to state carefully and prove the theorem of Section 8, establishing a correspondence between regular logic and bicategories of relations. The development will be detailed yet terse, as we have already explained the main ideas behind the theorem in Section 8.

We first define a formal system for regular logic. The syntax is borrowed from [Awo09] and the proof system from [Joh02]. We depart from the standard formulation of regular logic only by dispensing with function symbols. This convention is merely a convenience; Appendix B shows how to incorporate function symbols.

Definition. A *(multisorted) signature* consists of
- a set of *sorts* or *basic types*, which we write generically as $A, B, A_1, B_1, \ldots$, and
- a set of *relation symbols*, which we write as $R, S, \ldots$, where each relation symbol R is associated with a (possibly empty) ordered list of types $(A_1, \ldots, A_n)$.

For brevity, we often use the vector notation $\underline{A} := (A_1, \ldots, A_n)$. To express that relation symbol R has types $\underline{A}$, we write $R : \underline{A}$ or $R : (A_1, \ldots, A_n)$.

There is a countably infinite set of *variables* $x, y, z, \ldots$. Unlike in some typed logical calculi, the variable symbols do not have fixed types. Instead, we write $x : A$ to express that x is a variable of type A. A *context* is a (possibly empty) finite, ordered list of form

$$\Gamma = \underline{x} : \underline{A} = (x_1 : A_1, \ldots, x_n : A_n),$$

where the x_i's are distinct variables and the A_i's are basic types.

With respect to a fixed signature, the *formulas in context* of the regular language are expressions of form $\Gamma \,|\, \varphi$, where Γ is a context and φ is a formula, as defined inductively by the formation rules in Figure 1. Formulas outside of a context have no definite meaning. Note that the only *terms in context* of the regular language are variables, for which there is a single formation rule:

$$\Gamma, x : A, \Gamma' \,|\, x : A.$$

54

$$\frac{\Gamma \mid x_1 : A_1 \qquad \cdots \qquad \Gamma \mid x_n : A_n}{\Gamma \mid R(x_1, \ldots, x_n)} \qquad \text{(relation symbol } R : \underline{A})$$

$$\frac{\Gamma \mid x : A \qquad \Gamma \mid y : A}{\Gamma \mid x = y} \qquad \text{(equality)}$$

$$\frac{\Gamma \mid \varphi}{\Gamma, x : A \mid \varphi} \qquad \text{(weakening)}$$

$$\Gamma \mid \top \qquad \text{(truth)}$$

$$\frac{\Gamma \mid \varphi \qquad \Gamma \mid \psi}{\Gamma \mid \varphi \wedge \psi} \qquad \text{(conjunction)}$$

$$\frac{\Gamma, x : A \mid \varphi}{\Gamma \mid \exists x : A.\varphi} \qquad \text{(existential quantifier)}$$

Figure 1: Formation rules for regular logic

$$\varphi \vdash \varphi \qquad \text{(identity)}$$

$$\frac{\varphi \vdash \chi \qquad \chi \vdash \psi}{\varphi \vdash \psi} \qquad \text{(cut)}$$

$$\frac{\underline{x} : \underline{A} \mid \varphi \vdash \psi \qquad \Gamma \mid \underline{t} : \underline{A}}{\Gamma \mid \varphi[\underline{t}/\underline{x}] \vdash \psi[\underline{t}/\underline{x}]} \qquad \text{(substitution)}$$

$$\varphi \vdash x = x \qquad (\underline{x} = \underline{y}) \wedge \varphi \vdash \varphi[\underline{y}/\underline{x}] \qquad \text{(equality)}$$

$$\varphi \vdash \top \qquad \text{(truth)}$$

$$\frac{\varphi \vdash \psi \qquad \varphi \vdash \chi}{\varphi \vdash \psi \wedge \chi} \qquad \varphi \wedge \psi \vdash \varphi \qquad \varphi \wedge \psi \vdash \psi \qquad \text{(conjunction)}$$

$$\frac{\Gamma, x : A \mid \varphi \vdash \psi}{\Gamma \mid (\exists x : A.\varphi) \vdash \psi} \qquad \text{(existential quantifier)}$$

$$\Gamma \mid \varphi \wedge (\exists x : A.\psi) \vdash \exists x : A.(\varphi \wedge \psi) \qquad [x \notin \Gamma] \qquad \text{(Frobenius)}$$

Figure 2: Inference rules for regular logic

A *sequent* is an expression of form $\Gamma \mid \varphi \vdash \psi$, where $\Gamma \mid \varphi$ and $\Gamma \mid \psi$ are both formulas in context. The inference rules for sequents are listed in Figure 2.

The statement of the inference rules assumes the following notational conventions. Contexts that are constant across the premises and conclusion of a rule are omitted. The formulas in context appearing in the rules are implicitly assumed to be well-formed. For example, the existential quantifier rule assumes that x does not appear freely in ψ because $\Gamma \mid \psi$ must be well-formed. The vector notation $\underline{x} = \underline{y}$ is shorthand for $x_1 = y_1 \wedge \cdots \wedge x_n = y_n$; likewise, $\exists \underline{x} : \underline{A}$ is shorthand for $\exists x_1 : A_1 \cdots \exists x_n : A_n$. The expression $\varphi[\underline{y}/\underline{x}]$ denotes the simultaneous substitution of y_i for x_i, for $1 \le i \le n$, in the formula φ.

Remark. Several inference rules deserve further comment.
- *Substitution*: Useful special cases of the substitution rule include the *weakening* and *strengthening* rules

$$\frac{\Gamma \mid \varphi \vdash \psi}{\Gamma, x : A \mid \varphi \vdash \psi} \qquad \frac{\Gamma, x : A \mid \varphi \vdash \psi \qquad \Gamma \mid t : A}{\Gamma \mid \varphi \vdash \psi}.$$

- *Existential quantifier*: Given the other rules, the bidirectional existential quantifier rule is equivalent to the $\exists$-*introduction* and $\exists$-*elimination* rules [Jac99, Lemma 4.1.8]

$$\frac{\Gamma \mid t : A \qquad \Gamma \mid \varphi \vdash \psi[t/x]}{\Gamma \mid \varphi \vdash \exists x : A.\psi} \qquad \frac{\Gamma \mid \varphi \vdash \exists x : A.\psi \qquad \Gamma, x : A \mid \psi \vdash \chi}{\Gamma \mid \varphi \vdash \chi}.$$

- *Frobenius*: The so-called "Frobenius axiom," linking conjunction and existential quantification, is superfluous in full first-order logic with implication but is *not* deducible from the other rules of regular logic [Joh02, p. 831]. The converse rule *is* deducible [Joh02, p. 832]. The omission of the Frobenius rule in some standard texts on regular logic, such as [Oos95] and [But98], is apparently an error.

Definition. A *regular theory* (with respect to a fixed signature) is defined by a set of sequents in the signature, not necessarily finite, called the *axioms* of theory. Under a regular theory $\mathbb{T}$, a formula φ *entails* ψ, written $\Gamma \mid \varphi \vdash_{\mathbb{T}} \psi$, if the sequent $\Gamma \mid \varphi \vdash \psi$ is deducible from the axioms of $\mathbb{T}$ using the inference rules of regular logic (Figure 2). In this case we say that the sequent $\Gamma \mid \varphi \vdash \psi$ is an *entailment* or *theorem* of $\mathbb{T}$.

We now begin to establish the correspondence between bicategories of relations and regular logic by constructing the classifying category of a regular theory.

Definition. The *classifying category* of a regular theory $\mathbb{T}$, denoted $\mathrm{Cl}(\mathbb{T})$, is the bicategory of relations defined as follows. Its objects are finite lists of basic types $\underline{A} : (A_1, \ldots, A_n)$. Given a context $\Gamma = \underline{x} : \underline{A}$, we also write $[\Gamma] := \underline{A}$. Its morphisms $\underline{A} \to \underline{B}$ are equivalence classes of formulas in context,

$$[\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi],$$

where the equivalence relation $\sim$ is defined by

$$(\underline{x} : \underline{A}, \underline{y} : \underline{B} \mid \varphi) \sim (\underline{x}' : \underline{A}, \underline{y}' : \underline{B} \mid \varphi') \qquad \text{iff} \qquad \underline{x} : \underline{A}, \underline{y} : \underline{B} \mid \varphi \dashv\vdash_{\mathbb{T}} \varphi'[\underline{x}/\underline{x}', \underline{y}/\underline{y}'].$$

In other words, the morphisms of $\mathrm{Cl}(\mathbb{T})$ are formulas in context up to α-equivalence and logical equivalence under $\mathbb{T}$. Here $\dashv\vdash_{\mathbb{T}}$ is shorthand for $\dashv_{\mathbb{T}}$ and $\vdash_{\mathbb{T}}$, and the semicolon in a context $(\Gamma; \Gamma')$ is an extralogical marker that partitions the context into the domain

$[\Gamma]$ and codomain $[\Gamma']$ of the morphism $[\Gamma; \Gamma' \mid \phi]$. The 2-morphisms of $\mathrm{Cl}(\mathbb{T})$ are the entailments of $\mathbb{T}$:

$$[\Gamma; \Gamma' \mid \varphi] \implies [\Gamma; \Gamma' \mid \psi] \qquad \text{iff} \qquad \Gamma, \Gamma' \mid \varphi \vdash_{\mathbb{T}} \psi.$$

We now define the requisite structures to make the classifying category into a bicategory of relations. Composition of morphisms is given by

$$[\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi] \cdot [\underline{y} : \underline{B}; \underline{z} : \underline{C} \mid \psi] := [\underline{x} : \underline{A}; \underline{z} : \underline{C} \mid \exists \underline{y} : \underline{B}.\varphi \wedge \psi],$$

and the identity morphisms are

$$1_{\underline{A}} := [\underline{x} : \underline{A}; \underline{x}' : \underline{A} \mid \underline{x} = \underline{x}'].$$

The monoidal product is defined on objects by $\underline{A} \otimes \underline{B} := (\underline{A}, \underline{B})$ and on morphisms by

$$[\Gamma; \Delta \mid \varphi] \otimes [\Gamma'; \Delta' \mid \psi] := [\Gamma, \Gamma'; \Delta, \Delta' \mid \varphi \wedge \psi].$$

The monoidal unit is the empty list $I := ()$. The braidings are

$$\sigma_{\underline{A}, \underline{B}} := [\underline{x} : \underline{A}, \underline{y} : \underline{B}; \underline{y}' : \underline{B}, \underline{x}' : \underline{A} \mid (\underline{x} = \underline{x}') \wedge (\underline{y} = \underline{y}')].$$

Finally, the diagonals are

$$\Delta_{\underline{A}} := [\underline{x} : \underline{A}; \underline{x}' : \underline{A}, \underline{x}'' : \underline{A} \mid (\underline{x} = \underline{x}') \wedge (\underline{x} = \underline{x}'')]$$
$$\Diamond_{\underline{A}} := [\underline{x} : \underline{A} \mid \top].$$

Lemma. *The classifying category* $\mathrm{Cl}(\mathbb{T})$ *of a regular theory* $\mathbb{T}$ *is a bicategory of relations.*

Proof. We must check that every axiom of a bicategory of relations can be deduced from the inference rules of regular logic. The proofs are tedious but mechanical; we sketch a few to illustrate what is involved and leave the rest to the reader.

First, we show that $\mathrm{Cl}(\mathbb{T})$ is a category. To prove that composition is associative, we must show that the two formulas

$$\underline{x} : \underline{A}; \underline{w} : \underline{D} \mid \exists \underline{y} : \underline{B}.(\varphi \wedge \exists \underline{z} : \underline{C}.(\psi \wedge \chi))$$
$$\underline{x} : \underline{A}; \underline{w} : \underline{D} \mid \exists \underline{z} : \underline{C}.(\exists \underline{y} : \underline{B}.(\varphi \wedge \psi) \wedge \chi)$$

are equivalent in regular logic. In fact, both formulas are equivalent to

$$\underline{x} : \underline{A}; \underline{w} : \underline{D} \mid \exists \underline{y} : \underline{B}.\exists \underline{z} : \underline{C}.(\varphi \wedge \psi \wedge \chi).$$

The derivation relies crucially on the Frobenius rule. We omit the details and the proof of the identity axiom of a category.

Next, we prove that $\mathrm{Cl}(\mathbb{T})$ is a locally posetal 2-category. It is obvious that $\mathrm{Cl}(\mathbb{T})$ is locally posetal. The vertical composition axiom is immediate from the cut rule. To prove the horizontal composition axiom, suppose we have two entailments

$$\underline{x} : \underline{A}, \underline{y} : \underline{B} \mid \varphi \vdash \theta \qquad \text{and} \qquad \underline{y} : \underline{B}, \underline{z} : \underline{C} \mid \psi \vdash \chi.$$

By the conjunction and weakening rules,

$$\underline{x} : \underline{A}, \underline{y} : \underline{B}, \underline{z} : \underline{C} \mid \varphi \wedge \psi \vdash \theta \wedge \chi.$$

Using the $\exists$-introduction rule and then the bidirectional $\exists$-rule, we obtain

$$\underline{x} : \underline{A}, \underline{z} : \underline{C} \mid \exists \underline{y} : \underline{B}.(\varphi \wedge \psi) \vdash \exists \underline{y} : \underline{B}.(\theta \wedge \chi),$$

proving the validity of horizontal composition.

The axiom on lax comonoid homomorphisms amounts to two easily proved entailments, namely

$$\underline{x} : \underline{A}; y' : \underline{B}, y'' : \underline{B} \mid \exists \underline{y} : \underline{B}.(\varphi \wedge (\underline{y} = y') \wedge (\underline{y} = y'')) \vdash \varphi[y'/\underline{y}] \wedge \varphi[y''/\underline{y}]$$

and

$$\underline{x} : \underline{A} \mid \exists \underline{y} : \underline{B}.\varphi \vdash \top.$$

Finally, we mention that the adjoints of $\Delta_{\underline{A}}$ and $\Diamond_{\underline{A}}$ exist and are equal to

$$\nabla_{\underline{A}} := [\underline{x}' : \underline{A}, \underline{x}'' : \underline{A}; \underline{x} : \underline{A} \mid (\underline{x}' = \underline{x}) \wedge (\underline{x}'' = \underline{x})] \qquad \text{and} \qquad \Box_{\underline{A}} := [; \underline{x} : \underline{A} \mid \top]. \qquad \Box$$

Next, we construct the internal language of a bicategory of relations. A preliminary definition is:

Definition. An *interpretation* or *model* of a signature in a bicategory of relations $\mathcal{B}$ is specified by

- for every basic type A, an object $[\![A]\!]$ of $\mathcal{B}$;
- for every relation symbol $R : \underline{A}$, a morphism $[\![R]\!] : [\![\underline{A}]\!] \to I$ of $\mathcal{B}$, where we define $[\![\underline{A}]\!] := [\![A_1]\!] \otimes \cdots \otimes [\![A_n]\!]$.

If $\Gamma = \underline{x} : \underline{A}$ is a context, we also write $[\![\Gamma]\!] := [\![\underline{A}]\!]$.

An interpretation of a signature extends to the full regular language in that signature. By induction on the formation rules of regular logic (Figure 1), we assign to each formula in context $\Gamma, \Gamma' \mid \varphi$ a morphism $[\![\Gamma; \Gamma' \mid \varphi]\!] : [\![\Gamma]\!] \to [\![\Gamma']\!]$ of $\mathcal{B}$. In interpreting a rule, we allow ourselves to order the context variables and place the context semicolon however is most convenient, with the understanding that any other arrangement can be achieved by a suitable braiding and bending of wires. This flexibility greatly simplifies the notation.

- *Relation symbol*: Given a relation symbol $R : \underline{A}$,

$$[\![\underline{x} : \underline{A}, \Gamma \mid R(\underline{x})]\!] := [\![R]\!] \otimes \Diamond_{[\![\Gamma]\!]}.$$

- *Equality*: There are two cases: when the variables are distinct, set

$$[\![\Gamma, x : A; y : A \mid x = y]\!] := \Diamond_{[\![\Gamma]\!]} \otimes 1_{[\![A]\!]};$$

when the variables are equal, set

$$[\![\Gamma \mid x = x]\!] := \Diamond_{[\![\Gamma]\!]}.$$

- *Weakening*: Given a morphism $[\![\Gamma \mid \varphi]\!] : [\![\Gamma]\!] \to I$,

$$[\![\Gamma, x : A \mid \varphi]\!] := [\![\Gamma \mid \varphi]\!] \otimes \Diamond_{[\![A]\!]}.$$

- *Truth*: $[\![\Gamma; \Gamma' \mid \top]\!] := \top_{[\![\Gamma]\!], [\![\Gamma']\!]} = \lozenge_{[\![\Gamma]\!]} \square_{[\![\Gamma']\!]}$.
- *Conjunction*: Given morphisms $[\![\Gamma; \Gamma' \mid \varphi]\!] : [\![\Gamma]\!] \to [\![\Gamma']\!]$ and $[\![\Gamma; \Gamma' \mid \psi]\!] : [\![\Gamma]\!] \to [\![\Gamma']\!]$,

$$[\![\Gamma; \Gamma' \mid \varphi \wedge \psi]\!] := \Delta_{[\![\Gamma]\!]} \cdot ([\![\Gamma; \Gamma' \mid \varphi]\!] \otimes [\![\Gamma; \Gamma' \mid \psi]\!]) \cdot \nabla_{[\![\Gamma']\!]}.$$

- *Existential quantifier*: Given a morphism $[\![\Gamma; x : A \mid \varphi]\!] : [\![\Gamma]\!] \to [\![A]\!]$,

$$[\![\Gamma \mid \exists x : A.\varphi]\!] := [\![\Gamma; x : A \mid \varphi]\!] \cdot \lozenge_{[\![A]\!]}.$$

Definition. An interpretation $[\![\cdot]\!]$ of the signature of a regular theory $\mathbb{T}$ in a bicategory of relations $\mathcal{B}$ is an *interpretation* or *model* of $\mathbb{T}$ in $\mathcal{B}$ if it satisfies all the axioms of $\mathbb{T}$, i.e, for every axiom

$$\Gamma \mid \varphi \vdash \psi \quad \text{of} \quad \mathbb{T},$$

there is a 2-morphism

$$[\![\Gamma \mid \varphi]\!] \implies [\![\Gamma \mid \psi]\!] \quad \text{in} \quad \mathcal{B}.$$

The previous lemma can be interpreted as a *completeness* theorem, stating that the inference rules of regular logic are sufficient to prove every axiom of a bicategory of relations. The next lemma is a *soundness* theorem: it says that every inference rule of regular logic is valid in an arbitrary bicategory of relations.

Lemma. *Let $[\![\cdot]\!]$ be an interpretation of a regular theory $\mathbb{T}$ in a bicategory of relations $\mathcal{B}$. For every theorem*

$$\Gamma \mid \varphi \vdash_{\mathbb{T}} \psi \quad of \quad \mathbb{T},$$

there is a 2-morphism

$$[\![\Gamma \mid \varphi]\!] \implies [\![\Gamma \mid \psi]\!] \quad in \quad \mathcal{B}.$$

Proof. The proof is by induction on the derivation of a theorem of $\mathbb{T}$. By the definition of an interpretation, every axiom of $\mathbb{T}$ holds in $\mathcal{B}$. Therefore, it suffices to show that every inference rule of regular logic (Figure 2) is valid in $\mathcal{B}$. We sketch these proofs below.
- *Identity*: Existence of identity 2-morphism.
- *Cut*: Vertical composition of 2-morphisms.
- *Substitution*: Omitted.
- *Equality*: Omitted.
- *Truth*: By the lax comonoid homomorphism axiom,

$$[\![\Gamma \mid \varphi]\!] = [\![\Gamma \mid \varphi]\!] \cdot \lozenge_I \implies \lozenge_{[\![\Gamma]\!]} = [\![\Gamma \mid \top]\!].$$

- *Conjunction*: If $[\![\Gamma; \Gamma' \mid \varphi]\!] \Rightarrow [\![\Gamma; \Gamma' \mid \psi]\!]$ and $[\![\Gamma; \Gamma' \mid \varphi]\!] \Rightarrow [\![\Gamma; \Gamma' \mid \chi]\!]$, then

$$\begin{aligned}
[\![\Gamma; \Gamma' \mid \varphi]\!] &= \Delta_{[\![\Gamma]\!]}([\![\Gamma; \Gamma' \mid \varphi]\!] \otimes [\![\Gamma; \Gamma' \mid \varphi]\!]) \nabla_{[\![\Gamma']\!]} \\
&\Rightarrow \Delta_{[\![\Gamma]\!]}([\![\Gamma; \Gamma' \mid \psi]\!] \otimes [\![\Gamma; \Gamma' \mid \chi]\!]) \nabla_{[\![\Gamma']\!]} \\
&= [\![\Gamma; \Gamma' \mid \psi \wedge \chi]\!].
\end{aligned}$$

That proves the first conjunction rule. For the second, calculate

$$\begin{aligned}
[\![\Gamma; \Gamma' \mid \varphi \wedge \psi]\!] &= \Delta_{[\![\Gamma]\!]}([\![\Gamma; \Gamma' \mid \varphi]\!] \otimes [\![\Gamma; \Gamma' \mid \psi]\!]) \nabla_{[\![\Gamma']\!]} \\
&\Rightarrow \Delta_{[\![\Gamma]\!]}([\![\Gamma; \Gamma' \mid \varphi]\!] \otimes \lozenge_{[\![\Gamma]\!]} \square_{[\![\Gamma']\!]}) \nabla_{[\![\Gamma']\!]} \\
&= [\![\Gamma; \Gamma' \mid \varphi]\!].
\end{aligned}$$

The proof of the third rule is similar.

- *Existential quantifier*: Fix formulas $[\![\Gamma; x : A \mid \varphi]\!] : [\![\Gamma]\!] \to [\![A]\!]$ and $[\![\Gamma \mid \psi]\!] : [\![\Gamma]\!] \to I$. By the weakening formation rule, $[\![\Gamma; x : A \mid \psi]\!] = [\![\Gamma \mid \psi]\!] \cdot \square_{[\![A]\!]}$. If there is an entailment $[\![\Gamma; x : A \mid \varphi]\!] \Rightarrow [\![\Gamma; x : A \mid \psi]\!]$, then

$$
\begin{aligned}
[\![\Gamma \mid \exists x : A.\varphi]\!] &= [\![\Gamma; x : A \mid \varphi]\!]\, \Diamond_{[\![A]\!]} \\
&\Rightarrow [\![\Gamma; x : A \mid \psi]\!]\, \Diamond_{[\![A]\!]} \\
&= [\![\Gamma \mid \psi]\!]\, \square_{[\![A]\!]} \Diamond_{[\![A]\!]} \\
&= [\![\Gamma \mid \psi]\!].
\end{aligned}
$$

The proof of the converse rule is similar.

- *Frobenius*: Given formulas $[\![\Gamma \mid \varphi]\!] : [\![\Gamma]\!] \to I$ and $[\![\Gamma, x : A \mid \psi]\!] : [\![\Gamma]\!] \to [\![A]\!]$, compute

$$
\begin{aligned}
[\![\Gamma \mid \varphi \wedge (\exists x : A.\psi)]\!] &= \Delta_{[\![\Gamma]\!]}([\![\Gamma \mid \varphi]\!] \otimes [\![\Gamma \mid \exists x : A.\psi]\!]) \\
&= \Delta_{[\![\Gamma]\!]}([\![\Gamma \mid \varphi]\!] \otimes [\![\Gamma, x : A \mid \psi]\!]\, \Diamond_{[\![A]\!]}) \\
&= \Delta_{[\![\Gamma]\!]}([\![\Gamma \mid \varphi]\!]\, \square_{[\![A]\!]} \otimes [\![\Gamma, x : A \mid \psi]\!]) \nabla_{[\![A]\!]} \Diamond_{[\![A]\!]} \\
&= \Delta_{[\![\Gamma]\!]}([\![\Gamma, x : A \mid \varphi]\!] \otimes [\![\Gamma, x : A \mid \psi]\!]) \nabla_{[\![A]\!]} \Diamond_{[\![A]\!]} \\
&= [\![\Gamma; x : A \mid \varphi \wedge \psi]\!]\, \Diamond_{[\![A]\!]} \\
&= [\![\Gamma \mid \exists x : A.(\varphi \wedge \psi)]\!]. \qquad \square
\end{aligned}
$$

Definition. The *internal language* of a small bicategory of relations $\mathcal{B}$ is the regular theory $\mathrm{Lang}(\mathcal{B})$ defined as follows. Its signature consists of

- a basic type A for every object A of $\mathcal{B}$, and
- a relation symbol $R : (A_1, \ldots, A_n)$ for every morphism $R : A_1 \otimes \cdots \otimes A_n \to I$ of $\mathcal{B}$.

A sequent $\Gamma \mid \varphi \vdash \psi$ is an axiom of $\mathrm{Lang}(\mathcal{B})$ if and only if $[\![\Gamma \mid \varphi]\!] \Rightarrow [\![\Gamma \mid \psi]\!]$ in $\mathcal{B}$, where $[\![\cdot]\!]$ is the obvious interpretation of the signature of $\mathrm{Lang}(\mathcal{B})$ in $\mathcal{B}$.

By the lemma, $\mathrm{Lang}(\mathcal{B})$ is interpretable in $\mathcal{B}$ and the theorems of $\mathrm{Lang}(\mathcal{B})$ are exactly the 2-morphism of $\mathcal{B}$.

Remark. In general, a single morphism of $\mathcal{B}$ gives rise to many relation symbols of $\mathrm{Lang}(\mathcal{B})$, e.g., if $R : A \to I$ is morphism and $A = A_1 \otimes A_2$, then there are relation symbols $R : (A)$ and $R : (A_1, A_2)$.

We have now developed the machinery to state and prove the main theorem of Section 8.

Theorem. *For every small bicategory of relations $\mathcal{B}$, there is an equivalence of categories*

$$
\mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \simeq \mathcal{B} \qquad in \qquad \textbf{BiRel}.
$$

Proof. To prove the equivalence, it suffices to construct a structure-preserving functor $F : \mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \to \mathcal{B}$ that is full, faithful, and essentially surjective on objects [Rie16, Theorem 1.5.9]. Define the functor F on objects by

$$
F(\underline{A}) := [\![\underline{A}]\!] = [\![A_1]\!] \otimes \cdots \otimes [\![A_n]\!],
$$

where $\underline{A} = (A_1, \ldots, A_n)$ and each A_i is a basic type of $\mathrm{Lang}(\mathcal{B})$. If $\Gamma = \underline{x} : \underline{A}$ is a context, we also write $F(\Gamma) := [\![\Gamma]\!]$. Define F on morphisms $[\Gamma; \Gamma' \mid \varphi] : [\Gamma] \to [\Gamma']$ by

$$
F([\Gamma; \Gamma' \mid \varphi]) := [\![\Gamma; \Gamma' \mid \varphi]\!] : F(\Gamma) \to F(\Gamma').
$$

By the construction of the classifying category and the internal language, we have the fundamental equivalence

$$[\Gamma; \Gamma' \mid \varphi] \Rightarrow [\Gamma; \Gamma' \mid \psi] \text{ in } \mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \qquad \text{iff} \qquad \Gamma, \Gamma' \mid \varphi \vdash_{\mathrm{Lang}(\mathcal{B})} \psi$$

$$\text{iff} \qquad [\![\Gamma; \Gamma' \mid \varphi]\!] \Rightarrow [\![\Gamma; \Gamma' \mid \psi]\!] \text{ in } \mathcal{B}.$$

In particular, the functor F is well-defined and faithful. It is full because if $R : A \to B$ is a morphism of $\mathcal{B}$, then there exists a relation symbol $R : (A, B)$ of $\mathrm{Lang}(\mathcal{B})$ such that $F([\![x : A; y : B \mid R(x, y)]\!]) = R$. Clearly, F is (essentially) surjective on objects.

It remains to prove that F is a structure-preserving functor. The fundamental equivalence says that F preserves 2-morphisms. We must show that F also preserves composition, identities, monoidal products, and all the other structures of a bicategory of relations. We prove that F preserves composition and products of morphisms and omit the other straightforward verifications. First, given morphisms $[\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]$ and $[\underline{y} : \underline{B}; \underline{z} : \underline{C} \mid \psi]$ of $\mathrm{Cl}(\mathrm{Lang}(\mathcal{B}))$, calculate

$$
\begin{aligned}
F([\underline{x} &: \underline{A}; \underline{y} : \underline{B} \mid \varphi] \cdot [\underline{y} : \underline{B}; \underline{z} : \underline{C} \mid \psi]) \\
&= F([\underline{x} : \underline{A}; \underline{z} : \underline{C} \mid \exists \underline{y} : \underline{B}.(\varphi \wedge \psi)]) \\
&= [\![\underline{x} : \underline{A}; \underline{z} : \underline{C} \mid \exists \underline{y} : \underline{B}.(\varphi \wedge \psi)]\!] \\
&= (1_{[\![\underline{A}]\!]} \otimes \eta_{[\![\underline{C}]\!]}) \, [\![\underline{x} : \underline{A}, \underline{z} : \underline{C}; \underline{y} : \underline{B} \mid \varphi \wedge \psi]\!] \, \Diamond_{[\![\underline{B}]\!]} \\
&= (1_{[\![\underline{A}]\!]} \otimes \eta_{[\![\underline{C}]\!]})([\![\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]\!] \otimes [\![\underline{z} : \underline{C}; \underline{y} : \underline{B} \mid \psi]\!]) \nabla_{[\![\underline{B}]\!]} \Diamond_{[\![\underline{B}]\!]} \\
&= (1_{[\![\underline{A}]\!]} \otimes \eta_{[\![\underline{C}]\!]})([\![\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]\!] \otimes [\![\underline{y} : \underline{B}; \underline{z} : \underline{C} \mid \psi]\!]^{\dagger}) \epsilon_{[\![\underline{B}]\!]} \\
&= [\![\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]\!] \cdot [\![\underline{y} : \underline{B}; \underline{z} : \underline{C} \mid \psi]\!] \\
&= F([\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]) \cdot F([\underline{y} : \underline{B}; \underline{z} : \underline{C} \mid \psi]).
\end{aligned}
$$

Given morphisms $[\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]$ and $[\underline{z} : \underline{C}; \underline{w} : \underline{D} \mid \psi]$, calculate

$$
\begin{aligned}
F([\underline{x} &: \underline{A}; \underline{y} : \underline{B} \mid \varphi] \otimes [\underline{z} : \underline{C}; \underline{w} : \underline{D} \mid \psi]) \\
&= F([\underline{x} : \underline{A}, \underline{z} : \underline{C}; \underline{y} : \underline{B}, \underline{w} : \underline{D} \mid \varphi \wedge \psi]) \\
&= [\![\underline{x} : \underline{A}, \underline{z} : \underline{C}; \underline{y} : \underline{B}, \underline{w} : \underline{D} \mid \varphi \wedge \psi]\!] \\
&= \Delta_{(\underline{A}, \underline{C})}([\![\underline{x} : \underline{A}, \underline{z} : \underline{C}; \underline{y} : \underline{B}, \underline{w} : \underline{D} \mid \varphi]\!] \otimes [\![\underline{x} : \underline{A}, \underline{z} : \underline{C}; \underline{y} : \underline{B}, \underline{w} : \underline{D} \mid \psi]\!]) \nabla_{(\underline{B}, \underline{D})} \\
&= (\Delta_{\underline{A}} \otimes \Delta_{\underline{C}})(1_{\underline{A}} \otimes \sigma_{\underline{A}, \underline{C}} \otimes 1_{\underline{C}}) \\
&\qquad ([\![\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]\!] \otimes \Diamond_{\underline{C}} \Box_{\underline{D}} \otimes \Diamond_{\underline{A}} \Box_{\underline{B}} \otimes [\![\underline{z} : \underline{C}; \underline{w} : \underline{D} \mid \psi]\!]) \\
&\qquad (1_{\underline{B}} \otimes \sigma_{\underline{D}, \underline{B}} \otimes 1_{\underline{D}})(\nabla_{\underline{B}} \otimes \nabla_{\underline{D}}) \\
&= (\Delta_{\underline{A}} \otimes \Delta_{\underline{C}})([\![\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]\!] \otimes \Diamond_{\underline{A}} \Box_{\underline{B}} \otimes \Diamond_{\underline{C}} \Box_{\underline{D}} \otimes [\![\underline{z} : \underline{C}; \underline{w} : \underline{D} \mid \psi]\!])(\nabla_{\underline{B}} \otimes \nabla_{\underline{D}}) \\
&= [\![\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]\!] \otimes [\![\underline{z} : \underline{C}; \underline{w} : \underline{D} \mid \psi]\!] \\
&= F([\underline{x} : \underline{A}; \underline{y} : \underline{B} \mid \varphi]) \otimes F([\underline{z} : \underline{C}; \underline{w} : \underline{D} \mid \psi]).
\end{aligned}
$$

The first calculation is obscured by the wire bending needed to unpack the definitions, but becomes transparent when rendered as a string diagram. $\qquad \square$

Remark. The inverse functor $G : \mathcal{B} \to \mathrm{Cl}(\mathrm{Lang}(\mathcal{B}))$ in the equivalence is defined on objects by $G(A) := A$ and on morphisms $R : A \to B$ by $G(R) := [x : A; y : B \mid R(x, y)]$. As an alternate proof, it is possible to explicitly construct the natural isomorphisms $F \cdot G \cong 1_{\mathrm{Cl}(\mathrm{Lang}(\mathcal{B}))}$ and $G \cdot F \cong 1_{\mathcal{B}}$.

B. Coherent logic and distributive bicategories of relations

In this appendix we prove the main theorem of Section 9.3, establishing a correspondence between coherent logic with product and sum types and distributive bicategories of relations. We present the logical system carefully because we cannot find a comparable logic in the literature. However, where the proof overlaps with Appendix A, we provide less detail.

We first define a formal system for coherent logic with product and sum types. For the sake of brevity, we refer to this system simply as "coherent logic." We maintain the syntactic conventions of Appendix A, including the vector notation. Our proof system is an amalgamation of the usual proof systems for typed lambda calculus [Jac99; FDB06] and coherent logic [Joh02] but with several important differences that we shall point out.

Definition (cf. [Jac95; Jac99]). A *distributive signature* consists of
- a set of *basic types*, which generates a set of *types* according to the BNF grammar

$$A, B ::= C \mid A \times B \mid 1 \mid A + B \mid 0,$$

  where C ranges over basic types;
- a set of *function symbols* $f, g, h, \dots$, each with fixed domain type A and codomain type B, written $f : A \to B$;
- a set of *relation symbols* $R, S, \dots$, each with fixed domain types A and B, written $R : (A, B)$.

Variables and contexts are defined as in Appendix A. With respect to a fixed distributive signature, a *term in context* is an expression of form $\Gamma \mid t : A$, as defined inductively by the formation rules in Figure 3. Likewise, a *formula in context* is an expression $\Gamma \mid \varphi$ defined by the formation rules in Figure 4.

A *sequent* is an expression $\Gamma \mid \varphi \vdash \psi$, where $\Gamma \mid \varphi$ and $\Gamma \mid \psi$ are both formulas in context. The sequent $\Gamma \vdash \varphi$ is shorthand for $\Gamma \mid \top \vdash \varphi$. The inference rules for sequents are listed in Figure 5. As before, we omit the context when it is the same in the premises and conclusion of a rule.

Remark. We comment on the inference rules that differ from both first-order logic and equational type theory. Most importantly, product types are treated exactly as in type theory but sum types have a stronger axiomatization.
- *Distributivity*: Like the Frobenius rule, the distributivity axiom linking conjunction and disjunction is superfluous in full first-order logic but is *not* deducible from the other rules of coherent logic [Joh02, p. 831]. The converse of the distributivity axiom *is* deducible.
- *Case*: Our case rules cannot be expressed in simple type theory. Instead, type theory postulates

$$\vdash \delta\left(t, x : A.s[\iota_1(x)/z], y : B.s[\iota_2(y)/z]\right) = s[t/z].$$

  Given the other axioms, this rule can be deduced from our first case rule but not conversely.

$$\Gamma, x : A, \Gamma' \mid x : A \qquad \qquad \text{(variable)}$$

$$\frac{\Gamma \mid t : A}{\Gamma \mid f(t) : B} \qquad \qquad \text{(function symbol } f : A \to B)$$

$$\frac{\Gamma \mid t : A \qquad \Gamma \mid s : B}{\Gamma \mid \langle t, s \rangle : A \times B} \qquad \qquad \text{(pair)}$$

$$\frac{\Gamma \mid t : A \times B}{\Gamma \mid \pi_1(t) : A} \qquad \frac{\Gamma \mid t : A \times B}{\Gamma \mid \pi_2(t) : B} \qquad \text{(projection)}$$

$$\Gamma \mid * : 1 \qquad \qquad \text{(singleton)}$$

$$\frac{\Gamma \mid t : A + B \qquad \Gamma, x : A \mid r : C \qquad \Gamma, y : B \mid s : C}{\Gamma \mid \delta\,(t, x : A.r, y : B.s) : C} \qquad \text{(case)}$$

$$\frac{\Gamma \mid t : A}{\Gamma \mid \iota_1(t) : A + B} \qquad \frac{\Gamma \mid t : B}{\Gamma \mid \iota_2(t) : A + B} \qquad \text{(inclusion)}$$

Figure 3: Formation rules for terms of coherent logic

$$\frac{\Gamma \mid t : A \qquad \Gamma \mid s : B}{\Gamma \mid R(t, s)} \qquad \qquad \text{(relation symbol } R : (A, B))$$

$$\frac{\Gamma \mid t : A \qquad \Gamma \mid s : A}{\Gamma \mid t = s} \qquad \qquad \text{(equality)}$$

$$\frac{\Gamma \mid \varphi}{\Gamma, x : A \mid \varphi} \qquad \qquad \text{(weakening)}$$

$$\Gamma \mid \top \qquad \qquad \text{(truth)}$$

$$\Gamma \mid \bot \qquad \qquad \text{(falsity)}$$

$$\frac{\Gamma \mid \varphi \qquad \Gamma \mid \psi}{\Gamma \mid \varphi \wedge \psi} \qquad \qquad \text{(conjunction)}$$

$$\frac{\Gamma \mid \varphi \qquad \Gamma \mid \psi}{\Gamma \mid \varphi \vee \psi} \qquad \qquad \text{(disjunction)}$$

$$\frac{\Gamma, x : A \mid \varphi}{\Gamma \mid \exists x : A.\varphi} \qquad \qquad \text{(existential quantifier)}$$

Figure 4: Formation rules for formulas of coherent logic

- *Empty*: In simple type theory, the empty type is axiomatized by an *empty function* $\perp_A : A$ obeying the formation and inference rules

$$\frac{t : 0}{\perp_A(t) : A} \qquad \frac{t : 0 \qquad s : A}{\vdash \perp_A(t) = s} \ .$$

In particular, one can derive $\Gamma, x : 0 \mid r = s$ for any terms r, s. We strengthen that result to $\Gamma, x : 0 \mid \perp$ and discard the empty function $\perp_A$.

A *coherent theory* is defined analogously to a regular theory.

Definition. The *classifying category* of a coherent theory $\mathbb{T}$, denoted $\mathrm{Cl}(\mathbb{T})$, is the distributive bicategory of relations whose objects are the types of $\mathbb{T}$ (basic or compound); whose morphisms $A \to B$ are equivalence classes of formulas in context $[x : A, y : B \mid \varphi]$, where equivalence is up to α-equivalence and provable logical equivalence under $\mathbb{T}$; and whose 2-morphisms are the theorems of $\mathbb{T}$.

The structures of a distributive bicategory of relations are defined as follows.
- *Category*: Composition is defined by

$$[x : A, y : B \mid \varphi] \cdot [y : B, z : C \mid \psi] := [x : A, z : C \mid \exists y : B.(\varphi \wedge \psi)]$$

and the identity morphisms are $1_A := [x : A, x' : A \mid x = x']$.
- *Tensor*: On objects, $A \otimes B := A \times B$ and $I := 1$; on morphisms,

$$\begin{aligned}
[x_1 : A_1, & y_1 : B_1 \mid \varphi] \otimes [x_2 : A_2, y_2 : B_2 \mid \psi] \\
& := [x : A_1 \times A_2, y : B_1 \times B_2 \mid \\
& \qquad \exists x_1 : A_1. \exists y_1 : B_1.(\varphi \wedge \pi_1 x = x_1 \wedge \pi_1 y = y_1) \wedge \\
& \qquad \exists x_2 : A_2. \exists y_2 : B_2.(\psi \wedge \pi_2 x = x_2 \wedge \pi_2 y = y_2)] \\
& = [x : A_1 \times A_2, y : B_1 \times B_2 \mid \varphi[\pi_1 x / x_1, \pi_1 y / y_1] \wedge \psi[\pi_2 x / x_2, \pi_2 y / y_2]].
\end{aligned}$$

The braidings are

$$\sigma_{A,B} := [x : A \times B, y : B \times A \mid (\pi_1 x = \pi_2 y) \wedge (\pi_2 x = \pi_1 y)].$$

- *Cotensor*: On objects, $A \oplus B := A + B$ and $O := 0$; on morphisms,

$$\begin{aligned}
[x_1 : A_1, & y_1 : B_1 \mid \varphi] \oplus [x_2 : A_2, y_2 : B_2 \mid \psi] \\
& := [x : A_1 + A_2, y : B_1 + B_2 \mid \\
& \qquad \exists x_1 : A_1. \exists y_1 : B_1.(\varphi \wedge \iota_1 x_1 = x \wedge \iota_1 y_1 = y) \vee \\
& \qquad \exists x_2 : A_2. \exists y_2 : B_2.(\psi \wedge \iota_2 x_2 = x \wedge \iota_2 y_2 = y)].
\end{aligned}$$

The braidings are

$$\begin{aligned}
\boldsymbol{\sigma}_{A,B} := [x : A + B, & y : B + A \mid \\
& \exists x' : A. \exists y' : B.((\iota_1 x' = x \wedge \iota_2 x' = y) \vee (\iota_2 y' = x \wedge \iota_1 y' = y))].
\end{aligned}$$

- *Diagonals and codiagonals*:

$$\begin{aligned}
\Delta_A & := [x : A, y : A \times A \mid \langle x, x \rangle = y] \\
\Diamond_A & := [x : A, y : 1 \mid \top] \\
\blacktriangledown_A & := [x : A + A, y : A \mid \delta(x, x_1 : A.x_1, x_2 : A.x_2) = y] \\
\blacksquare_A & := [x : 0, y : A \mid \perp].
\end{aligned}$$

$$\varphi \vdash \varphi \qquad\qquad\qquad\qquad \text{(identity)}$$

$$\frac{\varphi \vdash \chi \qquad \chi \vdash \psi}{\varphi \vdash \psi} \qquad\qquad\qquad\qquad \text{(cut)}$$

$$\frac{\underline{x} : \underline{A} \mid \varphi \vdash \psi \qquad \Gamma \mid \underline{t} : \underline{A}}{\Gamma \mid \varphi[\underline{t}/\underline{x}] \vdash \psi[\underline{t}/\underline{x}]} \qquad\qquad\qquad \text{(substitution)}$$

$$\vdash x = x \qquad (\underline{x} = \underline{y}) \wedge \varphi \vdash \varphi[\underline{y}/\underline{x}] \qquad \underline{x} = \underline{y} \vdash t = t[\underline{y}/\underline{x}] \qquad \text{(equality)}$$

$$\vdash \pi_1(\langle t, s \rangle) = t \qquad \vdash \pi_2(\langle t, s \rangle) = s \qquad\qquad \text{(projection)}$$

$$\vdash \langle \pi_1(t), \pi_2(t) \rangle = t \qquad\qquad\qquad \text{(pair)}$$

$$\vdash \delta\left(\iota_1(t), x : A.r, y : B.s\right) = r[t/x]$$
$$\vdash \delta\left(\iota_2(t), x : A.r, y : B.s\right) = s[t/y] \qquad\qquad \text{(inclusion)}$$

$$\Gamma \mid \vdash (\exists x : A.\iota_1(x) = t) \vee (\exists x : B.\iota_2(x) = t)$$
$$\Gamma \mid (\exists x : A.\iota_1(x) = t) \wedge (\exists x : B.\iota_2(x) = t) \vdash \bot \qquad [x \notin \Gamma] \qquad \text{(case)}$$

$$\frac{\Gamma \mid t : 1}{\Gamma \mid \vdash t = *} \qquad \Gamma, x : 0 \mid \vdash \bot \qquad\qquad \text{(singleton \& empty)}$$

$$\varphi \vdash \top \qquad \bot \vdash \varphi \qquad\qquad\qquad \text{(truth \& falsity)}$$

$$\frac{\varphi \vdash \psi \qquad \varphi \vdash \chi}{\varphi \vdash \psi \wedge \chi} \qquad \varphi \wedge \psi \vdash \varphi \qquad \varphi \wedge \psi \vdash \psi \qquad \text{(conjunction)}$$

$$\frac{\varphi \vdash \chi \qquad \psi \vdash \chi}{\varphi \vee \psi \vdash \chi} \qquad \varphi \vdash \varphi \vee \psi \qquad \psi \vdash \varphi \vee \psi \qquad \text{(disjunction)}$$

$$\varphi \wedge (\psi \vee \chi) \vdash (\varphi \wedge \psi) \vee (\varphi \wedge \chi) \qquad\qquad \text{(distributivity)}$$

$$\frac{\Gamma, x : A \mid \varphi \vdash \psi}{\Gamma \mid (\exists x : A.\varphi) \vdash \psi} \qquad\qquad\qquad \text{(existential quantifier)}$$

$$\Gamma \mid \varphi \wedge (\exists x : A.\psi) \vdash \exists x : A.(\varphi \wedge \psi) \qquad [x \notin \Gamma] \qquad \text{(Frobenius)}$$

Figure 5: Inference rules for coherent logic

Lemma. *The classifying category* Cl($\mathbb{T}$) *of a coherent theory* $\mathbb{T}$ *is a distributive bicategory of relations.*

The proof of the lemma is monstrously long but similar in many respects to the completeness proof in Appendix A. To exemplify the new considerations posed by product and sum types, we prove a different, "obvious" fact about local unions in the classifying category. We expect this will exhaust the reader's appetite for such calculations. We then proceed directly to the construction of the internal language of a distributive bicategory of relations.

Proposition. *Local unions in the classifying category of a coherent theory are given by logical disjunction:*

$$[x : A, y : B \,|\, \varphi] \cup [x : A, y : B \,|\, \psi] = [x : A, y : B \,|\, \varphi \vee \psi].$$

Proof. In an abuse of notation, we write $[\varphi] := [x : A, y : B \,|\, \varphi]$ and $[\psi] := [x : A, y : B \,|\, \psi]$. By definition,

$$[\varphi] \cup [\psi] = \blacktriangle_A([\varphi] \oplus [\psi])\blacktriangledown_B.$$

First, use the inclusion rules and the first case rule to show that

$$\blacktriangle_A = [x : A, y : A + A \,|\, (\iota_1 x = y) \vee (\iota_2 x = y)]$$
$$\blacktriangledown_B = [x : B + B, y : B \,|\, (x = \iota_1 y) \vee (x = \iota_2 y)].$$

Therefore,

$$
\begin{aligned}
([\varphi] \oplus [\psi])\blacktriangledown_B = [x' : A + A, y : B \,|\, \exists y' : B + B.(\\
((\exists x : A.\exists \tilde{y} : B.(\varphi[\tilde{y}/y] \wedge \iota_1 x = x' \wedge \iota_1 \tilde{y} = y')) \vee \\
(\exists x : A.\exists \tilde{y} : B.(\psi[\tilde{y}/y] \wedge \iota_2 x = x' \wedge \iota_2 \tilde{y} = y'))) \\
\wedge (y' = \iota_1 y \vee y' = \iota_2 y))].
\end{aligned}
$$

Use the Frobenius rule and the distributivity of existential quantifiers over disjunctions to put the formula into prenex normal form, then distribute the conjunction over the disjunctions to put the body into disjunctive normal form:

$$
\begin{aligned}
([\varphi] \oplus [\psi])\blacktriangledown_B = [x' : A + A, y : B \,|\, \exists x : A.\exists \tilde{y} : B.\exists y' : B + B.(\\
(\varphi[\tilde{y}/y] \wedge \iota_1 x = x' \wedge \iota_1 \tilde{y} = y' \wedge \iota_1 y = y') \vee \\
(\varphi[\tilde{y}/y] \wedge \iota_1 x = x' \wedge \iota_1 \tilde{y} = y' \wedge \iota_2 y = y') \vee \\
(\psi[\tilde{y}/y] \wedge \iota_2 x = x' \wedge \iota_2 \tilde{y} = y' \wedge \iota_1 y = y') \vee \\
(\psi[\tilde{y}/y] \wedge \iota_2 x = x' \wedge \iota_2 \tilde{y} = y' \wedge \iota_2 y = y'))].
\end{aligned}
$$

The first and last disjuncts are handled by the injectivity of the inclusions (deducible from the inclusion rules), e.g.,

$$\iota_1 \tilde{y} = y' \wedge \iota_1 y = y' \vdash \iota_1 \tilde{y} = \iota_1 y \vdash \tilde{y} = y.$$

Eliminate the two cross terms using the second case rule, e.g.,

$$\iota_1 \tilde{y} = y' \wedge \iota_2 y = y' \vdash \iota_1 \tilde{y} = \iota_2 y \vdash \bot.$$

Upon simplification the result is

$$([\varphi] \oplus [\psi])\blacktriangledown_B = [x' : A + A, y : B \mid \exists x : A.((\varphi \wedge \iota_1 x = x') \vee (\psi \wedge \iota_2 x = x'))].$$

The second half of the calculation is very similar and yields

$$[\varphi] \cup [\psi] = \blacktriangle_A([\varphi] \oplus [\psi])\blacktriangledown_B = [x : A, y : B \mid \varphi \vee \psi]. \qquad \square$$

Definition. An *interpretation* or *model* of a distributive signature in a distributive bicategory of relations $\mathcal{B}$ is specified by

- for every basic type C, an object $\llbracket C \rrbracket$ of $\mathcal{B}$;
- for every function symbol $f : A \to B$, a morphism $\llbracket f \rrbracket : \llbracket A \rrbracket \to \llbracket B \rrbracket$ of $\mathrm{Map}(\mathcal{B})$;
- for every relation symbol $R : (A, B)$, a morphism $\llbracket R \rrbracket : \llbracket A \rrbracket \to \llbracket B \rrbracket$ of $\mathcal{B}$.

The extension of the interpretation to any type of the signature is implicit in the definition:

$$\llbracket A \times B \rrbracket = \llbracket A \rrbracket \otimes \llbracket B \rrbracket, \quad \llbracket 1 \rrbracket = I, \quad \llbracket A + B \rrbracket = \llbracket A \rrbracket \oplus \llbracket B \rrbracket, \quad \llbracket 0 \rrbracket = O.$$

We also maintain the convention that $\llbracket \underline{x} : \underline{A} \rrbracket := \llbracket \underline{A} \rrbracket := \llbracket A_1 \rrbracket \otimes \cdots \otimes \llbracket A_n \rrbracket$.

An interpretation of a distributive signature extends to the full coherent language in that signature by induction on the term and formula formation rules. Each term in context $\Gamma \mid t : A$ is interpreted as a morphism $\llbracket \Gamma \mid t \rrbracket : \llbracket \Gamma \rrbracket \to \llbracket A \rrbracket$ of $\mathrm{Map}(\mathcal{B})$ as follows.

- *Variable*: $\llbracket \Gamma, x : A, \Gamma' \mid x : A \rrbracket := \Diamond_{\llbracket \Gamma \rrbracket} \otimes 1_{\llbracket A \rrbracket} \otimes \Diamond_{\llbracket \Gamma' \rrbracket}$.
- *Function symbol*: Given a function symbol $f : A \to B$ and a map $\llbracket \Gamma \mid t : A \rrbracket$,

$$\llbracket \Gamma \mid f(t) : B \rrbracket := \llbracket \Gamma \mid t : A \rrbracket \cdot \llbracket f \rrbracket.$$

- *Pair*: Given maps $\llbracket \Gamma \mid t : A \rrbracket$ and $\llbracket \Gamma \mid s : B \rrbracket$,

$$\llbracket \Gamma \mid \langle t, s \rangle : A \times B \rrbracket := \Delta_{\llbracket \Gamma \rrbracket}(\llbracket \Gamma \mid t : A \rrbracket \otimes \llbracket \Gamma \mid s : B \rrbracket).$$

- *Projection*: Given a map $\llbracket \Gamma \mid t : A \times B \rrbracket$,

$$\llbracket \Gamma \mid \pi_1(t) : A \rrbracket := \llbracket \Gamma \mid t : A \times B \rrbracket \, (1_{\llbracket A \rrbracket} \otimes \Diamond_{\llbracket B \rrbracket})$$
$$\llbracket \Gamma \mid \pi_2(t) : B \rrbracket := \llbracket \Gamma \mid t : A \times B \rrbracket \, (\Diamond_{\llbracket A \rrbracket} \otimes 1_{\llbracket B \rrbracket}).$$

- *Singleton*: $\llbracket \Gamma \mid * : I \rrbracket := \Diamond_{\llbracket \Gamma \rrbracket}$.
- *Case*: Given maps $\llbracket \Gamma \mid t : A + B \rrbracket$, $\llbracket \Gamma, x : A \mid r : C \rrbracket$, and $\llbracket \Gamma, y : B \mid s : C \rrbracket$, define $\llbracket \Gamma \mid \delta(t, x : A.r, y : B.s) : C \rrbracket$ to be

$$\Delta_{\llbracket \Gamma \rrbracket} \cdot (1_{\llbracket \Gamma \rrbracket} \otimes \llbracket \Gamma \mid t : A + B \rrbracket) \cdot d_{\llbracket \Gamma \rrbracket, \llbracket A \rrbracket, \llbracket B \rrbracket} \cdot (\llbracket \Gamma, x : A \mid r \rrbracket \oplus \llbracket \Gamma, y : B \mid s \rrbracket) \cdot \blacktriangledown_{\llbracket C \rrbracket},$$

where $d_{X,Y,Z} : X \otimes (Y \oplus Z) \to (X \otimes Y) \oplus (X \otimes Z)$ is the family of distributivity isomorphisms in $\mathcal{B}$.

- *Inclusion*: Given maps $\llbracket \Gamma \mid t : A \rrbracket$ and $\llbracket \Gamma \mid s : B \rrbracket$,

$$\llbracket \Gamma \mid \iota_1(t) : A + B \rrbracket := \llbracket \Gamma \mid t : A \rrbracket \oplus \blacksquare_B$$
$$\llbracket \Gamma \mid \iota_2(s) : A + B \rrbracket := \blacksquare_A \oplus \llbracket \Gamma \mid s : B \rrbracket.$$

Each formula in context $\Gamma; \Gamma' \mid \varphi$ is interpreted as a morphism $\llbracket \Gamma; \Gamma' \mid \varphi \rrbracket : \llbracket \Gamma \rrbracket \to \llbracket \Gamma' \rrbracket$ of $\mathcal{B}$ as follows.

- *Relation symbol*: Given a relation symbol $R : (A, B)$ and two maps $[\![\Gamma \mid t : A]\!]$ and $[\![\Gamma \mid s : B]\!]$, define

$$[\![\Gamma \mid R(t, s)]\!] := \Delta_{[\![\Gamma]\!]}([\![\Gamma \mid t : A]\!]\, [\![R]\!] \otimes [\![\Gamma \mid s : B]\!])\epsilon_{[\![B]\!]}.$$

where, as usual, $\epsilon_{[\![B]\!]} = \nabla_{[\![B]\!]}\Diamond_{[\![B]\!]}$.

- *Equality*: Given maps $[\![\Gamma \mid t : A]\!]$ and $[\![\Gamma \mid s : A]\!]$,

$$[\![\Gamma \mid t = s]\!] := \Delta_{[\![\Gamma]\!]}([\![\Gamma \mid t : A]\!] \otimes [\![\Gamma \mid s : A]\!])\epsilon_{[\![A]\!]}$$

- *Falsity*: $[\![\Gamma; \Gamma' \mid \bot]\!] := \bot_{[\![\Gamma]\!], [\![\Gamma']\!]} = \blacklozenge_{[\![\Gamma]\!]}\blacksquare_{[\![\Gamma']\!]}$.
- *Disjunction*: Given morphisms $[\![\Gamma; \Gamma' \mid \varphi]\!] : [\![\Gamma]\!] \to [\![\Gamma']\!]$ and $[\![\Gamma; \Gamma' \mid \psi]\!] : [\![\Gamma]\!] \to [\![\Gamma']\!]$,

$$[\![\Gamma; \Gamma' \mid \varphi \vee \psi]\!] := \blacktriangle_{[\![\Gamma]\!]} \cdot ([\![\Gamma; \Gamma' \mid \varphi]\!] \oplus [\![\Gamma; \Gamma' \mid \psi]\!]) \cdot \blacktriangledown_{[\![\Gamma']\!]}.$$

The other formation rules—weakening, truth, conjunction, and existential quantifier—are interpreted exactly as in Appendix A.

An *interpretation* or *model* of a coherent theory in a distributive bicategory of relations is defined analogously to an interpretation of a regular theory in a bicategory of relations. There is also a corresponding soundness theorem.

Lemma. *Let $[\![\cdot]\!]$ be an interpretation of a coherent theory $\mathbb{T}$ in a distributive bicategory of relations $\mathcal{B}$. For every theorem*

$$\Gamma \mid \varphi \vdash_{\mathbb{T}} \psi \quad of \quad \mathbb{T},$$

there is a 2-morphism

$$[\![\Gamma \mid \varphi]\!] \implies [\![\Gamma \mid \psi]\!] \quad in \quad \mathcal{B}.$$

Proof. As before, we must show that every inference rule of coherent logic (Figure 5) is valid in $\mathcal{B}$. We sketch the proofs for the rules that have not already been treated in Appendix A.

- *Projection*: The first projection axiom $\vdash \pi_1 \langle t, s \rangle = t$ holds because

$$\Delta_X (f \otimes g)\pi_{Y,Z} = \Delta_X (f \otimes g)(1_Y \otimes \Diamond_Z) = \Delta_X (f \otimes g\Diamond_Z) = \Delta_X (f \otimes \Diamond_X) = f$$

  whenever $f : X \to Y$ and $g : X \to Z$ are maps. The second projection axiom $\vdash \pi_2 \langle t, s \rangle = s$ is proved similarly.

- *Pair*: The pair axiom $\vdash \langle \pi_1(t), \pi_2(t) \rangle = t$ amounts to the equality

$$\begin{aligned}
\Delta_{X \otimes Y}(\pi_{X,Y} \otimes \pi'_{X,Y}) &= (\Delta_X \otimes \Delta_Y)(1_X \otimes \sigma_{X,Y} \otimes 1_Y)(1_X \otimes \Diamond_Y \otimes \Diamond_X \otimes 1_Y) \\
&= (\Delta_X \otimes \Delta_Y)(1_X \otimes \sigma_{X,Y}\Diamond_{Y \otimes X} \otimes 1_Y) \\
&= (\Delta_X \otimes \Delta_Y)(1_X \otimes \Diamond_X \otimes \Diamond_Y \otimes 1_Y) \\
&= 1_X \otimes 1_Y = 1_{X \otimes Y}.
\end{aligned}$$

- *Inclusion*: The two inclusion axioms are dual to the two projection axioms, e.g., $\iota_{X,Y}(f \oplus g)\blacktriangledown_Z = f$ for any two morphisms $f : X \to Z$ and $g : Y \to Z$ (which need not be maps).

- *Case*: The first case axiom $\vdash (\exists x : A.\iota_1 x = t) \vee (\exists x : B.\iota_2 x = t)$ amounts to

$$\square_{X \oplus Y} = \blacktriangle_I (\square_X \iota_{X,Y} \oplus \square_Y \iota'_{X,Y}) \blacktriangledown_{X \oplus Y}.$$

By the dual of the pair axiom, $(\iota_{X,Y} \oplus \iota'_{X,Y}) \blacktriangledown_{X \oplus Y} = 1_{X \oplus Y}$, it suffices to show that $\square_{X \oplus Y} = \blacktriangle_I (\square_X \oplus \square_Y)$, or equivalently

$$\lozenge_{X \oplus Y} = (\lozenge_X \oplus \lozenge_Y) \blacktriangledown_I =: [\lozenge_X, \lozenge_Y]$$

Because the inclusions are maps, we have $\iota_{X,Y} \lozenge_{X \oplus Y} = \lozenge_X$ and $\iota'_{X,Y} \lozenge_{X \oplus Y} = \lozenge_Y$, and hence the last equation holds by the universal property of the coproduct. We omit the proof of the second case axiom.
- *Singleton*: The unit I of the tensor is terminal in $\mathrm{Map}(\mathcal{B})$.
- *Empty*: For any objects X, Y of $\mathcal{B}$, there is at most one morphism $X \otimes O \to Y$, which must therefore be $\bot_{X \otimes O, Y}$. A similar result holds, with essentially the same proof, in a distributive category; we refer to [CLW93, Proposition 3.2].
- *Falsity*: Dual to truth.
- *Disjunction*: Dual to conjunction.
- *Distributivity*: As mentioned in Section 9.2, the distributive law

$$R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$$

holds in any distributive bicategory of relations. The proof, which we omit, is a calculation involving the canonical distributivity isomorphism. $\qquad \square$

Definition. The *internal language* of a small distributive bicategory of relations $\mathcal{B}$ is the coherent theory $\mathrm{Lang}(\mathcal{B})$ defined as follows. Its signature consists of
- for every object A of $\mathcal{B}$, a basic type A;
- for every pair of types (A, B) and every morphism $f : [\![A]\!] \to [\![B]\!]$ of $\mathrm{Map}(\mathcal{B})$, a function symbol $f : A \to B$; and
- for every pair of types (A, B) and every morphism $R : [\![A]\!] \to [\![B]\!]$ of $\mathcal{B}$, a relation symbol $R : (A, B)$.

A sequent $\Gamma \mid \varphi \vdash \psi$ is an axiom of $\mathrm{Lang}(\mathcal{B})$ if and only if $[\![\Gamma \mid \varphi]\!] \Rightarrow [\![\Gamma \mid \psi]\!]$ in $\mathcal{B}$. Throughout the definition, $[\![\cdot]\!]$ is the obvious interpretation of the signature of $\mathrm{Lang}(\mathcal{B})$ in the category $\mathcal{B}$.

Remark. The expressivity of the internal language is not affected by including function symbols for the maps because every map is also associated with a relation symbol.

As in Appendix A, a single morphism of $\mathcal{B}$ can give rise to many function and relation symbols of $\mathrm{Lang}(\mathcal{B})$. Moreover, despite replacing lists of types by compound types, it remains the case that $\mathrm{Lang}(\mathcal{B})$ has "too many" types. For example, if $A = A_1 \otimes A_2$ is a product in $\mathcal{B}$, then both A and $A_1 \times A_2$ are types in $\mathrm{Lang}(\mathcal{B})$. Although the types are isomorphic, they are formally distinct. This discrepancy explains why the equivalence of categories in the theorem below is not an isomorphism of categories.

Theorem. *For every small distributive bicategory of relations $\mathcal{B}$, there is an equivalence of categories*

$$\mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \simeq \mathcal{B} \qquad in \qquad \mathbf{DistBiRel}.$$

Proof. It suffices to construct a structure-preserving functor $F : \mathrm{Cl}(\mathrm{Lang}(\mathcal{B})) \to \mathcal{B}$ that is full, faithful, and essentially surjective on objects. Define the functor F on objects by $F(A) := [\![A]\!]$ and on morphisms by

$$F([x : A; y : B \mid \varphi]) := [\![x : A; y : B \mid \varphi]\!] : F(A) \to F(B).$$

The proof that F is well-defined and has the requisite properties proceeds as in Appendix A. We leave the details to the reader. $\qquad\square$