ELSEVIER

Journal of Pure and Applied Algebra 149 (2000) 213–239

JOURNAL OF PURE AND APPLIED ALGEBRA

www.elsevier.com/locate/jpaa

A categorical interpretation of C.S. Peirce's propositional logic Alpha

Geraldine Brady[a], Todd H. Trimble[b,*]

[a] *University of Chicago, Chicago, IL 60637, USA*

[b] *Department of Mathematical and Computer Science, Loyola University-Chicago, Chicago IL 60626, USA*

Received 28 October 1998

Communicated by P.J. Freyd

Abstract

C.S. Peirce's graphical system Alpha for propositional logic is given a geometric representation in terms of isotopy classes of planar diagrams, and surgery rules on these diagrams called illative transformations. An algebraic representation theorem is proven, stating that Alpha, as an equational theory, is isomorphic to the theory of Boolean algebras. The geometric and algebraic representations are extended to give multi-sorted analogues of Alpha. An interpretation of Alpha is given in terms of linear logic and the theory of closed categories. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 18C10; 18D15; 03G30; 03G05; 19D23; 01A55

1. Introduction

Toward the close of the 19th century, Charles S. Peirce developed a diagrammatic calculus for doing logic, which he called "existential graphs". Although Peirce wrote that he considered this calculus his *chef d'oeuvre*, he never published it, with one minor exception [19], and as mathematics it has since remained an obscure topic. So far the literature on existential graphs has come mostly from philosophers and workers in artificial intelligence and information theory [20, 21, 24, 23, 5, 1]. We believe a mathematical investigation is now in order.

The reasons why Peirce did not publish this work appear to have been partly political [11, 6], but perhaps the main reason was that existential graphs were considerably more

* Corresponding author.

E-mail address: trimble@math.luc.edu (T.H. Trimble).

PII: S0022-4049(98)00179-0

difficult and expensive to print in Peirce's day than the linear notation which became standard in logic.

Due to modern advances in computer graphics and typesetting systems, the standard linear notation for logic no longer carries a compelling advantage over a planar notation. Since Peirce's time, other graphical schemes have been developed for particle physics and general relativity (namely, Feynman diagrams and Penrose diagrams), enabling physicists to perform complicated calculations with speed and facility. Apparently, Peirce saw similar advantages in his graphs.

More recently, Joyal and Street [8] have begun putting the Feynman and Penrose diagrams on a firm mathematical basis, using the language of monoidal categories. As we intend to show, Peirce's work on existential graphs also fits in a categorical framework, and its more developed aspects (especially his system Beta; see below) resonate closely with the use of string diagrams in mathematics and physics. This may seem remarkable, but it should be borne in mind that Peirce had made a deep study of the calculus of relations, and certainly perceived a structure which today we would call a monoidal bicategory of relations. And in developing a syntax of graphs for the relational calculus (where the graphs were inspired in part by analogies with chemical valences and chemical bonds), he was in retrospect developing a syntax for monoidal bicategories which are freely generated in a suitable sense, which is precisely the type of situation addressed by the Joyal–Street calculus of string diagrams.

Existential graphs come in three parts. The first, which is the topic of this paper, is called Alpha and corresponds to propositional logic. The main theorem of this paper is a representation theorem which shows that Alpha, as an equational theory, is isomorphic to the theory of Boolean algebras. The second, which will be the topic of a forthcoming paper, is called Beta and corresponds to the first-order predicate calculus. More exactly, it corresponds to a representation of first-order logic by means of fibered categories, where the fibers are Boolean algebras and where quantification is represented by adjoints to pullback operations on the fibers. The third, which was quite speculative and incomplete by Peirce's own admission, is called Gamma and corresponds, we believe, to type-theoretic higher-order logic, as found in the study of freely generated Boolean toposes.

This paper is organized as follows. In Section 2, we first formulate in topological terms the notion of alpha graph as given by Peirce, and then give an equivalent combinatorial formulation in terms of algebraic theories. The notion of algebraic theory is central to the paper; part of Section 2 is devoted to generalities thereon. In Section 3, we present Peirce's notion of inference between alpha graphs, called illative transformation; the rules of inference are schemata which involve functorial constructions called context operators. From the rules of inference we obtain equations on alpha graphs, thus leading to a new equational or algebraic theory called Alpha. In Section 4 we give a precise comparison between the theory Alpha and the theory of Boolean algebras; our representation theorem states that the two theories are isomorphic. In Section 5, we place Peirce's system within a broader categorical context; this includes comparison with the linear logic of Girard and the theory of closed categories, which can be used

to give a conceptual explanation behind Peirce's illative transformations. In Section 6, we extend the one-sorted theory Alpha to multi-sorted or typed variants, thus laying groundwork for our treatment of Peirce's more sophisticated system Beta for first-order logic. This will be the subject of a forthcoming paper.

2. Alpha graphs

2.1. Geometric representations

We begin with a geometric definition of alpha graphs which is a modern formulation of the description given by Peirce [19]:

Definition 1. A pre-alpha graph on n variables $x_1,\dots,x_n$ is a planar configuration consisting of a disjoint union of "nodes" (points labeled by elements of $\{x_1,\dots,x_n\}$) and "seps" (short for separation lines, which are simple closed curves).

Alpha graphs will be defined as isotopy-equivalence classes of pre-alpha graphs. Our first task is to topologize the space of pre-alpha graphs.

The nodes of a pre-alpha graph are given by a finite subset F of the plane $\mathbf{R}^2$, together with a labeling function $F \to \{x_1,\dots,x_n\}$. The seps of a pre-alpha graph are represented by a topological embedding of a disjoint union of circles

$$\phi : \underbrace{(S^1 \cup \cdots \cup S^1)}_{k} \to \mathbf{R}^2.$$

Both nodes and seps are viewed as elements of unordered collections.

Thus, let $\mathrm{Map}(S^1,\mathbf{R}^2)$ be the space of continuous maps $S^1 \to \mathbf{R}^2$, with the compact-open topology. The topological embedding ϕ given above may be viewed as a point in the product space $\mathrm{Map}(S^1,\mathbf{R}^2)^k$. The symmetric group S_k acts on this product space by permuting components, and it acts freely on the subspace E_k of such embeddings ϕ. By passing to the orbit space of E_k under this action (i.e., the quotient space $Q_k = E_k/S_k$), we obtain the space of (unordered) sep configurations.

Similarly, the space of unordered node configurations may be viewed as an orbit space. We start with the collection of injective functions

$$f : \{p_1,\dots,p_j\} \to \mathbf{R}^2,$$

viewed as a subspace of $(\mathbf{R}^2)^j$, with the evident S_j-action. A node-labeling function may be represented by a function $\lambda : \{p_1,\dots,p_j\} \to \{x_1,\dots,x_n\}$, i.e., as a point in

$$\{x_1,\dots,x_n\}^j$$

with the evident S_j-action, where $\{x_1,\dots,x_n\}$ is given the discrete topology. Now the space C_j of configurations on j nodes is a subspace of the orbit space of

$$(\mathbf{R}^2)^j \times \{x_1,\dots,x_n\}^j$$

under the S_j-action $\sigma(f,\lambda) = (\sigma f, \sigma\lambda)$.

Thus, the space of pre-alpha graphs is defined to be the subspace X of

$$\sum_{j,k\geq 0} C_j \times Q_k,$$

where a pair of node and sep configurations belongs to X if and only if these configurations do not intersect in $\mathbf{R}^2$. We say that two pre-alpha graphs $p,q \in X$ are isotopic if there is a path between them in X.

Definition 2. An alpha graph is an isotopy class of pre-alpha graphs, i.e., an element of the set $\pi_0(X)$ of path components of X.

To avoid excessive verbiage, we will typically state things using pre-alpha graphs when we really intend alpha graphs, without bothering to mention isotopy-equivalence classes each time.

Notice that alpha graphs may be constructed by recursion (see also [11]), as follows:

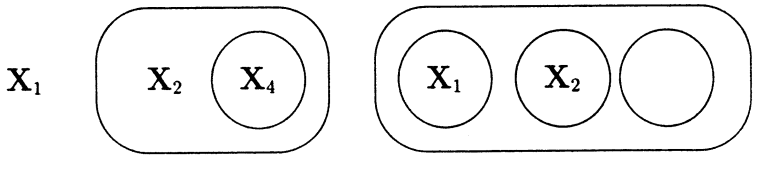
1. A node is an alpha graph.

2. The union of a sep together with an alpha graph G in its interior is an alpha graph (we refer to the interior containing G as a sep region, and we refer to this operation as "sepping" G). This is depicted as

3. A disjoint union of a finite (possible empty) collection of alpha graphs is an alpha graph. So

is an alpha graph if G_1 and G_2 are alpha graphs. By "disjoint union" of a set of graphs $\{G_i\}$, we mean both that G_i and G_j have empty intersection if $i \neq j$, and also in this case that every node and sep line of G_i is exterior to every sep line of G_j, and vice versa.

We observe that a given label x_i can occur more than once or not at all. For example, in the alpha graph



x_1 and x_2 occur twice and x_3 does not occur.

Alpha graphs on $x_1,\ldots,x_n$ should be thought of as corresponding to expressions in a propositional language, where we think of disjoint union as conjunction and sepping as

negation. Thus, the graph above corresponds to the Boolean expression $x_1 \wedge \neg(x_2 \wedge \neg x_4) \wedge \neg(\neg x_1 \wedge \neg x_2 \wedge \neg\top)$, where $\top$ ("true") is the empty conjunction corresponding to the empty sep region.

We have adopted Peirce's shorthand for (pre-)alpha graphs, which indicates the presence of nodes by their labels without giving the specific locations of their points. From the point of view of isotopy-equivalence, all we really care about is whether a node is interior or exterior to a sep line, and this is what the shorthand serves to identify.

Conjunction (i.e., disjoint union) of alpha graphs is a well-defined operation in an obvious way, as is negation (sepping). For example, if we wish to conjoin two graphs G and H, we apply an isotopy to move G outside of a neighborhood containing H, and then take their disjoint union. A familiar argument from elementary homotopy theory ensures that conjunction of alpha graphs is associative and commutative.

2.2. Alpha graphs as an algebraic theory

The recursive procedure we gave for constructing alpha graphs suggests that more abstract or combinatorial descriptions of alpha graphs are possible. One possibility is that an alpha graph be specified by something like a rooted tree, where each leaf of the tree is labeled by a primitive alpha graph (i.e., a node or an empty graph), and every other vertex is labeled either $\neg$ (if there is one incoming edge and one outgoing edge) or $\wedge_n$ (if there are n incoming edges and one outgoing edge, for $n>1$: $\wedge_n$ represents the n-fold conjunction).

In a moment we will give a similar abstract combinatorial description, related to an operation which is crucial to Peirce's calculus, namely, substitution of one graph in another. Roughly speaking, if G and H are alpha graphs and p is a point of H, one substitutes G for p in H by first finding an open neighborhood around p which intersects no other node or sep line of H, then erasing p, and then placing a copy of G in that neighborhood, thus obtaining a new graph $H[G/p]$.

Similar types of substitution operations occur throughout mathematics, particularly in universal algebra and in the theory of operads [18]. The types of substitution needed for the present paper can be defined for any monadic functor $U : A \to \mathcal{S}et$ [16], and in particular for the underlying set functor U on the category A of models of an equational theory, i.e., any one-sorted first-order theory described by finitary operations and axioms are which are universally quantified equations (disallowing disjunctions, negations, and existential quantifiers). The main construction needed to define substitution is that of a free A-algebra on a finite set.

For the sake of concreteness, let us consider substitution in the theory of Boolean algebras. Let B_n denote the free Boolean algebra on n indeterminates $x_1, \ldots, x_n$. There is a substitution map

$$B_k \times B_{n_1} \times \cdots \times B_{n_k} \to B_{n_1+\cdots+n_k}$$
$$\langle p; \vec{q} = \langle q_1, \ldots, q_k \rangle \rangle \mapsto p[\vec{q}/\vec{x}\,]$$

which, roughly speaking, substitutes Boolean polynomials q_j for variables x_j in p. We construct this map in two steps.

- First, we view $B_{n_1+\cdots+n_k}$ as isomorphic, in the category of Boolean algebras, to the coproduct (i.e., tensor product) $B_{n_1} \otimes \cdots \otimes B_{n_k}$, so that for $1 \leq j \leq k$ there is a canonical coproduct injection

$$i_j : B_{n_j} \to B_{n_1+\cdots+n_k}.$$

This injection sends $x_i \in B_{n_j}$ to its re-indexing as the variable $x_{s_{j+i}} \in B_{n_1+\cdots+n_k}$, where $s_j = \sum_{l=1}^{j-1} n_l$. We thus obtain a map

$$B_k \times B_{n_1} \times \cdots \times B_{n_k} \xrightarrow{1\times i_1\times\cdots\times i_k} B_k \times (B_{n_1+\cdots+n_k})^k.$$

- Second, if B is any Boolean algebra, there is a map

$$\begin{aligned} B_k \times B^k &\to B \\ \langle p, b_1, \ldots, b_k \rangle &\mapsto p(\vec{b}/\vec{x}), \end{aligned}$$

natural in B, where $p(\vec{b}/\vec{x})$ is the value of $p \in B_k$ under the unique Boolean homomorphism $B_k \to B$ which sends $x_j \in B_k$ to $b_j \in B$ for $j = 1, \ldots, k$. Specializing to $B = B_{n_1+\cdots+n_k}$, substitution is defined as the evident composite

$$B_k \times B_{n_1} \times \cdots \times B_{n_k} \to B_k \times (B_{n_1+\cdots+n_k})^k \to B_{n_1+\cdots+n_k}.$$

This construction can be generalized from Boolean algebras to any equational variety. From the second part of this construction, elements of a free algebra F_k on a set $\{x_1, \ldots, x_k\}$ (with respect to a given variety) are in natural bijection with the k-ary operations. In one direction, elements $p \in F_k$ induce operations of the form $X^k \to X$; in the other direction, a k-ary operation, applied to $X = F_k$, sends $\langle x_1, \ldots, x_k \rangle \in X^k$ to some $p \in X$, where $\{x_1, \ldots, x_k\}$ is the set which generates $X = F_k$. These correspondences between operations and elements in free algebras are mutually inverse. Thus, in the categorical approach to universal algebra, the basic insight is that there is a one-to-one correspondence between "syntax" (definable operations in an equational theory) and "semantics" (elements in free models).

Notice that this description of operations in terms of free algebras is independent of any particular presentation of the equational theory. This is another basic insight, due to Lawvere [13], in the categorical approach to universal algebra. Going from semantics to syntax, one defines the category of operations (or Lawvere algebraic theory) of a finitary equational variety as the opposite of the category of finitely generated free algebras. As a category, an algebraic theory admits finite products. Going from syntax to semantics, one defines a model of an algebraic theory $\mathbf{A}$ as a product-preserving functor $\mathbf{A} \to \mathscr{S}et$, and a model homomorphism as a natural transformation between such functors. The free models then correspond precisely to the representable functors on $\mathbf{A}$.

Definition 3 (*Lawvere*; *see Manes* [17])**.** An algebraic theory is a category with products for which every object is a power (n-fold product) of a distinguished object x.

The simplicity of this notion, and its freedom from presentation-dependent descriptions, has numerous advantages. For example, a *translation* between two algebraic theories **A** and **B** is nothing but a product-preserving functor $\mathbf{A} \to \mathbf{B}$.

Some remarks are in order. It is customary to denote the power x^n simply as n, so that every morphism in an algebraic theory **A** is of the form $n \xrightarrow{f} m$. Thus the categorical product $x^m \times x^n \cong x^{m+n}$ is denoted instead by a sum $m+n$. By the universal property of products, each morphism $n \xrightarrow{f} m$ is given by an m-tuple of morphisms $n \xrightarrow{f_i} 1$, where f_i is defined by composing f with the ith projection:

$$n \xrightarrow{f} m \xrightarrow{\pi_i} 1.$$

A morphism $n \to 1$ is called an n-ary operation of the theory; examples include the projections $n \xrightarrow{\pi_i} 1$. (These projections correspond to generators x_i in free algebras.) We can also define substitution by means of composition: for $n = n_1 + \cdots + n_k$, the composite

$$n \xrightarrow{q=\langle q_1,\ldots,q_k\rangle} k \xrightarrow{p} 1$$

corresponds to our $p[\vec{q}/\vec{x}]$. This composition is also used to explain the sense in which the representable $\hom(j,-)$ can be interpreted as an algebra: the underlying set of this algebra is $X = \hom(j,1)$, and the action by k-ary operations

$$\hom(k,1) \times X^k \to X$$

is given precisely by composition

$$\hom(k,1) \times \hom(j,k) \to \hom(j,1).$$

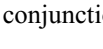
Following a standard abuse of language, we refer to such an algebra by its underlying set $\hom(j,1)$.

Now we apply these considerations to alpha graphs. Let $\mathscr{A}_k$ be the set of alpha graphs on $x_1,\ldots,x_k$. If $H \in \mathscr{A}_k$ and $G_j \in \mathscr{A}_{n_j}$ for $j = 1,\ldots,k$, we define $H[\vec{G}/\vec{x}]$ by first re-indexing the variables of the G_j in the manner described earlier for Boolean algebras, and then substituting the graph G_j for each occurrence of the label x_j in H, thus giving a map

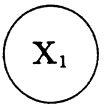
$$\mathscr{A}_k \times \mathscr{A}_{n_1} \times \cdots \times \mathscr{A}_{n_k} \to \mathscr{A}_{n_1+\cdots+n_k}.$$

Notice that every alpha graph can be constructed by starting with nodes ("projections") and the following graphs (primitive "operations"), and closing under substitution.

1. conjunction: $2 \xrightarrow{\wedge} 1$:



2. negation: $1 \xrightarrow{\neg} 1$:



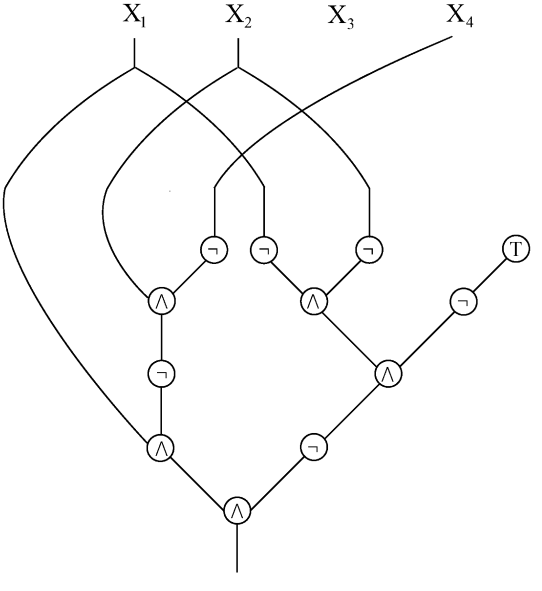
3. truth: $0 \xrightarrow{\top} 1$:

[empty graph]

Example. The alpha graph displayed at the end of Section 2.1 can be constructed from projections $\pi_i : 4 \to 1$ and the primitives $2 \xrightarrow{\wedge} 1$, $1 \xrightarrow{\neg} 1$, and $0 \xrightarrow{\top} 1$ as the following composite:

$$4 \xrightarrow{f} 5 \xrightarrow{2+3\neg+\top} 6 \xrightarrow{1+2\wedge+\neg} 4 \xrightarrow{1+\neg+\wedge} 3 \xrightarrow{\wedge+\neg} 2 \xrightarrow{\wedge} 1,$$

where $f = \langle \pi_1, \pi_2, \pi_4, \pi_1, \pi_2 \rangle$. To depict this information a tree notation is often useful:

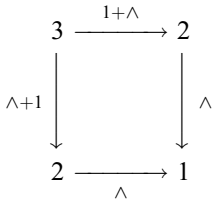


This leads to the combinatorial description of alpha graphs promised earlier.

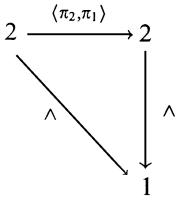
Definition 4. An alpha graph on n variables is an operation $n \to 1$ of the algebraic theory $\mathscr{A}$ generated by operations $2 \xrightarrow{\wedge} 1$, $1 \xrightarrow{\neg} 1$, and $0 \xrightarrow{\top} 1$, subject to the

associativity, commutativity, and unital equations:

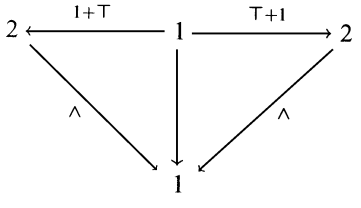
1. $(3 \xrightarrow{\wedge+1} 2 \xrightarrow{\wedge} 1) = (3 \xrightarrow{1+\wedge} 2 \xrightarrow{\wedge} 1)$:



2. $(2 \xrightarrow{\langle \pi_2, \pi_1 \rangle} 2 \xrightarrow{\wedge} 1) = (2 \xrightarrow{\wedge} 1)$:



3. $(1 \xrightarrow{1+\top} 2 \xrightarrow{\wedge} 1) = \mathrm{id}_1 = (1 \xrightarrow{\top+1} 2 \xrightarrow{\wedge} 1)$:



The set of operations $\hom(n, 1)$ is denoted $\mathscr{A}_n$.

3. Transformation rules

3.1. Contexts

In what follows, we freely pass back and forth between the geometric and combinatorial descriptions of alpha graphs. The next definition refers to the geometric description. It will be used to describe, in terms of substitution operations, the geometric operation referred to earlier as $H[G/p]$.

Definition 5. Suppose $G \in \mathscr{A}_n$ and $1 \leq i \leq n$. G is an x_i-context if the label x_i occurs exactly once in G. It is an x_i-constant if the label x_i does not occur.

This notion of "constant" makes sense for any algebraic theory. For example, in the theory of commutative rings, a binary operation corresponds to an element of $\mathbf{Z}[x_1, x_2]$, and x_1^2 would be an example of an x_2-constant.

The notion of "context" can also be generalized, but in a way which depends on a chosen generators-and-relations presentation of an algebraic theory. This generalization is implicit in the following lemma: it describes how an x_i-context can be constructed from disjoint union and sep, which generate the theory of alpha graphs.

Lemma 1. *Every x_i-context in $\mathscr{A}_n$ can be built up, starting with x_i and closing under two types of operations*:

1. *conjunction with an x_i-constant*;
2. *sepping* (*i.e.*, *negation*).

This lemma gives a recursive definition of context. The same definition can be applied to provide contexts for the theory of Boolean algebras, under the presentation where the primitive operations are taken to be conjunction and negation.

The number of sep lines which surround x_i in an x_i-context is independent of the order in which the context is built (from the operations given in the lemma). If the number of such seppings is even, the context is called *covariant*, and if the number is odd, the context is called *contravariant*.

If $H \in \mathscr{A}_n$ is an x_i-context, then the graph H has a single node p with label x_i. We have now an unary operator $H[-/x_i]$ on the algebra $\mathscr{A}_n$ which sends an alpha graph G to the graph we denoted in the beginning of Section 2.2 by $H[G/p]$: $H[-/x_i]$ substitutes G for x_i in H, and x_j for x_j whenever $j \neq i$. This operator will be abbreviated to H^* whenever the intended context variable x_i is clear. We observe for future reference that the operators H^* are closed under composition: that in fact $H^* \circ G^* = (H^*(G))^*$.

3.2. Illative transformation

For the purposes of his blackboard calculus, Peirce introduced what he calls "illative transformation", a procedure which replaces one alpha graph G by another graph H (or infers H from G) according to certain rules of inference. We view these rules as the specification of an entailment relation on alpha graphs, denoted $\vdash$. As we will see, it corresponds exactly to the entailment relation $a \vdash b$ on the corresponding Boolean expressions, but in a highly nontraditional manner.

There are three rules for illative transformation. In stating these rules, only alpha graphs belonging to some fixed $\mathscr{A}_n$ are to be considered.

Let p be an x_i-context, q an x_j-context, and let p^* (and similarly q^*) be the unary operators obtained by substituting for x_i (or x_j) inside p (or q). Then for alpha graphs A and B:

(1) *iteration* $q^*(p^*(A) \wedge B) \vdash q^*(p^*(A \wedge B) \wedge B)$;
de-iteration $q^*(p^*(A \wedge B) \wedge B) \vdash q^*(p^*(A) \wedge B)$.

(2) *weakening* $p^*(A \wedge B) \vdash p^*(A)$ if p is covariant;
$p^*(A) \vdash p^*(A \wedge B)$ if p is contravariant.
(3) *double sep introduction* $p^*(A) \vdash p^*(\neg\neg A)$;
double sep elimination $p^*(\neg\neg A) \vdash p^*(A)$.

These rules are used to generate a reflexive and transitive relation $\vdash$ on each $\mathscr{A}_n$: we take the smallest reflexive and transitive relation $\vdash$ which contains pairs (G,H) where G transforms to H. Taking alpha graphs as objects and instances of $A \vdash B$ as morphisms, we obtain category structures on the $\mathscr{A}_n$ in which there is at most one morphism between a given source and target; such categories are called *preorders*.

Using the fact that context operators p^* are closed under composition, it is immediate from the structure of the rules that if G transforms to H, then $p^*(G)$ transforms to $p^*(H)$ whenever p^* is covariant, and $p^*(H)$ transforms to $p^*(G)$ whenever p^* is contravariant. Hence the operators p^* are functorial on the $\mathscr{A}_n$.

The first rule is the most powerful, and perhaps the most mysterious rule of Peirce's system Alpha. Iteration says that if we have a graph B outside of a context p and A is a graph inside the context p, then we can insert a copy of the graph B next to A inside the context p, and the same can also be carried out within an ambient context q. De-iteration is the companion rule to iteration, and says that the transformation can be performed in the reverse direction. Our explanation for the iteration and de-iteration rules will be based on the theory of closed categories.

The next two rules are more ordinary. The second rule, weakening, intuitively means that if a context p^* is covariant, then we can think of graphs A and B inside this context as conclusions of an implication, and then weaken the implication by erasing one of them. If p^* is contravariant, then dually we can think of graphs inside p^* as hypotheses of an implication, and weaken by adding in an extra graph inside p^*.

The third rule, double sep introduction and elimination, says that double negation is the identity, i.e., if A is inside a context p, we can add or erase two sep lines around A.

Let us define graphs A and B to be illatively equivalent, denoted $A \sim B$, if $A \vdash B$ and $B \vdash A$. Clearly, the first and third rules can each be condensed into a single equivalence:

$$q^*(p^*(A) \wedge B) \sim q^*(p^*(A \wedge B) \wedge B);$$

$$p^*(A) \sim p^*(\neg\neg A).$$

Passing to equivalence classes, there are poset structures on the sets $\mathscr{A}_n / \sim$ induced from the preorder structures on the $\mathscr{A}_n$.

Moreover, we have the following lemma which ensures that composition or substitution of $\sim$-equivalence classes of alpha graphs $n \to 1$ in $\mathscr{A}$ is well-defined. Thus there is an algebraic theory $\alpha = \mathscr{A} / \sim$, a quotient of the algebraic theory $\mathscr{A}$ of alpha graphs. The set of morphisms $\hom_\alpha(n,1) = \mathscr{A}_n / \sim$ will be denoted by α_n.

Lemma 2. *If* $p, p' : k \to 1$ *are alpha graphs and* $p \sim p'$, *then* $p \circ q \sim p' \circ q$ *for all* $q : n \to k$. *If* $q = \langle q_1, \ldots q_k \rangle$, $q' = \langle q'_1, \ldots q'_k \rangle$, *and* $q_j \sim q'_j$ *whenever* $1 \leq j \leq k$, *then* $p \circ q \sim p \circ q'$ *for all* $p : k \to 1$.

Proof. Instances of $A \sim B$ are just isomorphisms in the preorders $\mathscr{A}_n$, so the lemma says that $-\circ q$ and $p\circ -$ are functorial with respect to subcategories of isomorphisms. The first statement is clear, and in fact $-\circ q$ preserves all entailments $p \vdash p'$ (is functorial with respect to all morphisms $p \vdash p'$), since for each transformation rule, application of $-\circ q$ merely involves replacing a graph A or B by a graph of type $A[\vec{q}/\vec{x}]$ or $B[\vec{q}/\vec{x}]$, thus giving another instance of the same rule. As for the second statement, even though composition $p\circ -$ may not preserve all entailments, a morphism $p\circ q$ can be constructed by means of a series of substitutions $[q_j/x_j]$ in p, one j at a time, and each such substitution can be viewed as an application of some context operator p_j^* to q_j. Depending on whether p_j is covariant or contravariant, p_j^* preserves or reverses entailments, but in either case it preserves instances of illative equivalence $q_j \sim q_j'$. The proof is complete.

4. Representation theorem

Now we state our main theorem:

Theorem 1. *The theory α is isomorphic to the theory β of Boolean algebras.*

The proof proceeds as follows. Clearly, both α and β are quotient theories of the theory of alpha graphs, since both theories are generated by conjunction, negation, and truth, and both satisfy the associativity, commutativity, and unital equations. For alpha graphs A and B, we write $A \sim_\alpha B$ if A and B are identified in α (i.e., if $A \sim B$), and $A \sim_\beta B$ if they are identified in β. The theorem is proven once we establish that $\sim_\alpha = \sim_\beta$ (as equivalence relations on the set of alpha graphs).

Thus the proof naturally decomposes into two parts:

1. *Completeness*: $\sim_\beta \subseteq \sim_\alpha$. This says that the equations for Boolean algebras are all derivable from the equations for α, or more simply, that each poset $\alpha_n = \hom_\alpha(n,1)$ is a Boolean algebra (recall that a poset can be a Boolean algebra in at most one way).

2. *Soundness*: $\sim_\alpha \subseteq \sim_\beta$. This says that the illative transformation rules for Alpha are derivable from the Boolean algebra axioms.

Completeness: First, α_n is a meet-semilattice. For this it suffices to show that $A \vdash B$ in α if and only if $A = A \wedge B$ (whence idempotence of $\wedge$ follows). If $A \vdash B$, then $A \vdash A \wedge A \vdash A \wedge B$ where the first transformation follows by iteration; also we have $A \wedge B \vdash A$ by weakening, so $A = A \wedge B$ follows. Conversely, from $A = A \wedge B$ and the weakening $A \wedge B \vdash B$ we obtain $A \vdash B$.

Next, we observe that negation, being a contravariant involution, must transform finite meets into finite joins, and vice versa. Thus α_n is a lattice. If we show this lattice is a Heyting algebra, we are done since Boolean algebras are exactly Heyting algebras in which double negation is the identity.

Thus, we define the implication $B \to C$ as $\neg(B \wedge \neg C)$ and show $A \wedge B \vdash C$ if and only if $A \vdash (B \to C)$. It clearly suffices to prove modus ponens $(B \to C) \wedge B \vdash C$ and its

"dual" $A \vdash B \to (A \wedge B)$. To prove modus ponens, we observe

$$\neg(B \wedge \neg C) \wedge B \vdash \neg(\neg C) \wedge B \vdash B,$$

where the first transformation is de-iteration and the second is weakening. To prove the dual, we observe

$$A \vdash \neg(\neg A) \vdash \neg(B \wedge \neg A) \vdash \neg(B \wedge \neg(A \wedge B)),$$

where the first transformation is double sep introduction, the second is weakening, and the third is iteration. Thus completeness is established.

Soundness: The weakening and double sep introduction/elimination rules for α follow trivially from their counterparts in the theory β of Boolean algebras, so it remains to derive the iteration and de-iteration rules within β. This is not really difficult, since context operators p^* are built recursively from instances of $C \wedge -$ and negation, so that it suffices to establish the rules for these special cases.

As our main purpose is to give a broader categorical meaning to Peirce's calculus, we give a quick proof of this now; then we explain at length the deeper conceptual significance of the iteration and de-iteration rules.

For $p^* = \neg$, we have

$$\neg(A \wedge B) \wedge B = (\neg A \vee \neg B) \wedge B = (\neg A \wedge B) \vee (\neg B \wedge B) = \neg A \wedge B.$$

For $p^* = C \wedge -$, we have

$$(C \wedge (A \wedge B)) \wedge B = (C \wedge A) \wedge B$$

trivially, since $\wedge$ is idempotent. The proof of soundness is complete.

Corollary 1. *The poset α_n of alpha graphs modulo illative equivalence is isomorphic to the free Boolean algebra on n generators.*

5. Categorical interpretations

As our main theorem shows, Peirce's system Alpha is equivalent to classical propositional logic. In this section, we will explore the relationship between Alpha and other forms of propositional logic, especially linear logic [3] and its attendant categorical aspects [22]. As we shall see, the categorical notion of strength can be used to give a conceptual explanation for Peirce's iteration and de-iteration rules.

Peirce's original idea was that one could quickly establish tautologies in propositional logic by means of a graphical blackboard calculus. Since that time, there has been intensive study on whether there exist feasible algorithms for establishing such tautologies, and related study on the complexity theory aspects of the Gentzen cut elimination result for propositional sequent calculus. Such study reveals the critical role played by the structural rules in the sequent calculus, and especially the contraction

rule: the act of pushing cuts past contractions can produce an exponential explosion in the size of proofs. It also reveals the indeterminancy of the cut elimination algorithm: the Church–Rosser property is not satisfied.

Motivated in part by these considerations, Girard introduced linear logic, in which the contraction and weakening rules in their global form are dropped or, more accurately, are admitted in a controlled way through the use of exponential modal operators. On the categorical side, the contraction rule for conjunction corresponds to the presence of diagonal maps $\Delta : X \to X \times X$, and the weakening rule to the presence of projection maps $\pi : X \times Y \to X$. By dropping these structures, one opens up the possibility of more general tensor products $\otimes$ playing the role of the conjunction operation. The unit I for such a tensor product then plays the role of the truth value "true".

Thus, linear logic is intimately related to monoidal categories [14] or, more exactly, to closed symmetric monoidal categories. The presence of symmetry isomorphisms $\sigma_{xy} : X \otimes Y \to Y \otimes X$ corresponds to the exchange rule in sequent calculus, and "closed" refers to the existence of exponential objects C^B which play the role of implications $B \to C$. Thus, in a closed monoidal category, there is a natural bijection

$$\hom(A \otimes B, C) \cong \hom(A, C^B)$$

which plays the role of what we earlier referred to as "modus ponens" and its "dual".

A $*$-autonomous category is a closed symmetric monoidal category which is equipped with a suitable "negation" operator. One assumes given an object D (the dualizing object) which plays the role of the truth value "false". Defining negation as the contravariant functor D^{-}, one can construct a natural map (the double-dual embedding)

$$\eta_A : A \to D^{D^A}.$$

The axiom is that this is a natural *isomorphism*: the double negation functor is isomorphic to the identity functor. Thus, $*$-autonomous categories model a form of *classical* propositional logic where the contraction and weakening rules are dropped; similarly, closed symmetric monoidal categories can be viewed as modeling a form of *intuitionistic* propositional logic.

Remark. There are various species belonging to the genus of "closed categories": closed monoidal, closed symmetric monoidal, $*$-autonomous, cartesian closed, etc., each associated with some fragment of propositional logic. Their importance goes far beyond the possibility of doing generalized logic. Indeed, closed category theory is more properly regarded as an abstract theory of functions and functional composition. But it interacts with other functional theories like typed λ-calculus and combinatory logic, and these interactions feed back into logic in interesting ways, showing for example how proofs of a sequent $A \vdash B$ may be regarded as abstract functions $A \to B$ (see [4], especially its discussion of Heytingian versus Tarskian semantics). One way to do this is to start with a syntactic construction of a free closed category on a set

of generators $\{x_1,\ldots,x_n\}$, whose morphisms are represented by formal concatenations or directed paths of basic structural arrows or entailments, modulo basic structural equations and naturality equations. Then one can set up a map which sends formulas A of the associated sequent calculus to objects A of the free structure, and sequent deductions of $A \vdash B$ to morphisms $A \to B$ of the free structure. A "completeness theorem" states that such a map is onto (the map from formulas to objects is a bijection). One defines two deductions of $A \vdash B$ to be "the same" (or Lambek-equivalent) if they map to the same morphism $A \to B$ [12]. In this way, one can identify free closed categories with categories whose morphisms are (equivalence classes of) sequent deductions.

Boolean algebras are (highly specialized) examples of $*$-autonomous categories, where the morphisms are instances of $a \leq b$ and the monoidal product is conjunction. Conversely, we have the following proposition (due to Joyal): a $*$-autonomous category in which the monoidal product is the ordinary (cartesian) product is a preorder which is equivalent to a Boolean algebra, in the sense that the induced poset of isomorphism classes (as α_n is induced by $\mathscr{A}_n$) is isomorphic to a Boolean algebra.

Proof. Since negation in a $*$-autonomous category transforms coproducts into products, and vice versa, it follows that the dualizing object D is the empty coproduct, i.e., the initial object 0. If A is any object, then $0 \times A \cong 0$ by the distributive law ($- \times A$ preserves coproducts because it is left adjoint to $-^A$). Given a morphism $A \to 0$, one then concludes that $A \cong 0$ because A is then a retract of $0 \times A \cong 0$. It follows that there can be at most one arrow $A \to 0$. It now follows that for any two objects A and B, there is at most one arrow $B \to A$, since such arrows correspond to arrows $B \to 0^{0^A}$, or to arrows $B \times 0^A \to 0$, and we just saw there is at most one of these. Hence, the category is a preorder, and so the products and coproducts here are meets and joins. In other words, the associated poset is a lattice, and has a Heyting algebra structure on account of the tensor-hom adjunction, and thus a Boolean algebra structure by the isomorphism $A \cong 0^{0^A}$, thus completing the proof.

Remark. Cartesian $*$-autonomous categories are the categorical models appropriate for classical propositional logic. One may proceed as in the preceding remark and give a syntactic construction of the free model on $\{x_1,\ldots,x_n\}$, by taking directed paths of Alpha transformations, and passing to equivalence classes of paths given by appropriate naturality and structural requirements. We will come to these requirements presently when we discuss the notion of "strength". But the preceding proposition shows that the resulting structure is a preorder; in fact, it is equivalent to the preorder α_n of alpha graphs, which we constructed using a much simpler method. Thus there are no interesting categorical semantics for the propositional Gentzen sequent calculus besides Boolean algebras. In other words, any two sequent deductions of a sequent $A \vdash B$ are the same in the sense of Lambek-equivalence: any two "functions" $A \to B$ in this system

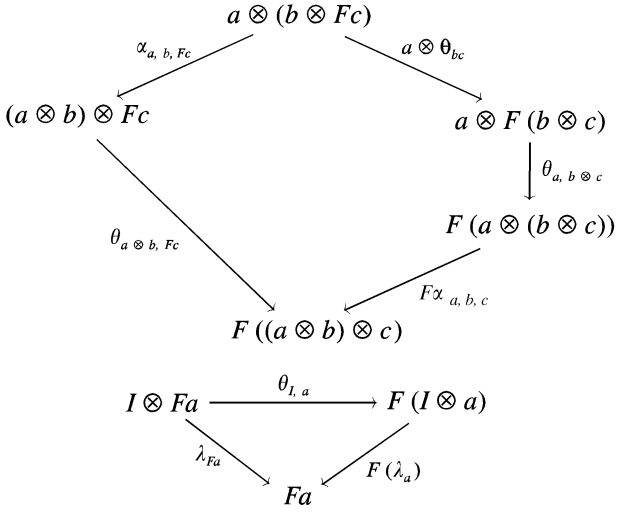
are provably equal, a phenomenon which in the context of λ-calculus might be called "algorithmic inconsistency" [4].

Thus, in order to put Alpha in a broader categorical context, it will be necessary to decompose the rules along the lines suggested by linear logic. This applies especially to iteration and de-iteration, which are amalgams of contraction/weakening rules with the more linear or monoidal notion of strength.

Definition 6 (*Kelly* [10])**.** Let $(V,\otimes,I)$ be a monoidal category, with associativity α, left unit λ, and right unit ρ. Given a covariant functor $F:V\to V$, a (left) strength for F is a natural transformation

$$a\otimes F(b)\xrightarrow{\theta_{ab}}F(a\otimes b)$$

such that the following diagrams commute:



Examples. For any object c in a monoidal category V, a canonical strength for $F=-\otimes c$ is given by the associativity isomorphism $\alpha_{abc}:a\otimes(b\otimes c)\to(a\otimes b)\otimes c$. If V is symmetric monoidal, then $F=c\otimes-$ carries a strength obtained by conjugating the strength for $-\otimes c$ by the symmetry: $-\otimes c\to c\otimes-$. If V is closed symmetric monoidal, then the covariant functors $-^c:V\to V$ carry a strength

$$a\otimes b^c\to(a\otimes b)^c$$

obtained by applying the hom-tensor adjunction to the composite

$$(a\otimes b^c)\otimes c\to a\otimes(b^c\otimes c)\to a\otimes b,$$

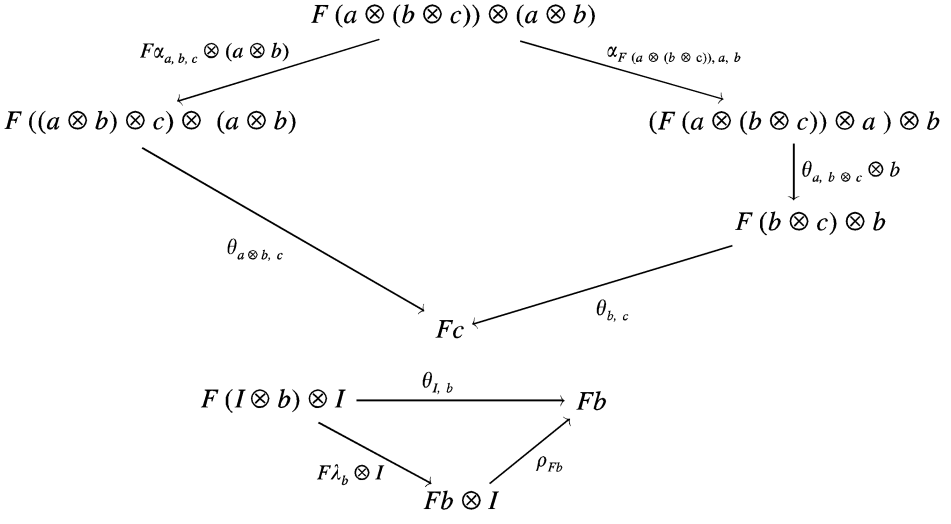
where the first map is the inverse of an associativity map, and the second is $a \otimes -$ applied to an evaluation map.

Now we define the notion of strength for a contravariant functor:

Definition 7. Given a monoidal category V and a contravariant functor $F : V \to V$, a (right) strength is an extranatural transformation

$$F(a \otimes b) \otimes a \xrightarrow{\theta_{ab}} F(b)$$

such that the following diagrams commute:



Example. If V is closed symmetric monoidal and c is an object of V, then $F = c^-$ carries a canonical strength, obtained by applying the hom-tensor adjunction to the composite

$$(c^{a \otimes b} \otimes a) \otimes b \to c^{a \otimes b} \otimes (a \otimes b) \to c,$$

where the first map is the inverse of an associativity map, and the second is an evaluation map.

From the theory of closed categories, we have the following general constructions: if V is closed symmetric monoidal, then strengths on $F : V \to V$ are in canonical bijection with V-functor structures on F. In one direction, if F is V-functorial, then F carries a canonical strength obtained by applying the hom-tensor adjunction to the composite

$$a \to (a \otimes b)^b \to F(a \otimes b)^{F(b)},$$

where the first map is a co-evaluation map and the second is an instance of the structure map for V-functoriality. In the other direction, given a strength on F, we obtain a V-functor structure on F by applying the hom-tensor adjunction to

$$a^b \otimes F(b) \to F(a^b \otimes b) \to F(a),$$

where the first map is an instance of strength and the second is F applied to an evaluation. Similarly, and in a V-dual sense, strengths on contravariant functors $F : V \to V$ are in canonical bijection with V-functor structures on F.

If V is closed symmetric monoidal, then under the usual composition of V-functors, the corresponding strengths also compose in an evident way. For example, if $F, G : V \to V$ are strong contravariant endofunctors on V, then there is a strong covariant endofunctor $F \circ G : V \to V$ on V. Similar statements hold for other combinations of covariant/contravariant endofunctors. In fact, strong functors $V \to V$ of either variance compose in an evident way if we assume only that V is symmetric monoidal; we leave the verification to the reader. Also, the distinction between left and right strengths disappears in the presence of a symmetry.

In particular, let us consider the case where $V = V_n$ is the free closed symmetric monoidal category on n generators $x_1, \ldots, x_n$.

Remark. In this case, "free" refers to a left adjoint F to the underlying functor U from the category of closed symmetric monoidal categories to the category of categories. The morphisms in the former category are strict closed functors: functors preserving closed structure strictly, i.e., preserving $\otimes$, associativities, symmetries, unit maps, exponentiation, evaluations, and co-evaluations. The free model on n generators refers to the value under F of the discrete category whose objects are $x_1, \ldots, x_n$.

We can define the set of x_i-contexts in $V = V_n$ as the closure of x_i under the class of V-functors $C \otimes -, - \otimes C$, C^-, and $-^C$, where C is an x_i-constant in V_n (an object or formula with no occurrence of the variable x_i). Each x_i-context p induces a corresponding (covariant or contravariant) context V-functor $p^* : V_n \to V_n$, defined by substituting for x_i in p. More precisely, p^* is the functor which sends an object B to the value of p under the unique strict closed functor $V_n \to V_n$ that maps x_i to B, and x_j to x_j if $j \neq i$. Each of these V-functors p^* is V-strong according to the foregoing discussion. Thus:

1. For p covariant, there is a strength $p^*(A) \otimes B \to p^*(A \otimes B)$.
2. For p contravariant, there is a strength $p^*(B \otimes A) \otimes B \to p^*(A)$.

These structural maps (1) and (2) suggest illative transformation rules for a linear logic form of Alpha. Let p and q be contexts in $\mathscr{A}_n$, and let A, B be graphs in $\mathscr{A}_n$. Then

(L1) If p is covariant, then $q^*(p^*(A) \wedge B) \vdash q^*(p^*(A \wedge B))$ if q is covariant, and $q^*(p^*(A \wedge B)) \vdash q^*(p^*(A) \wedge B)$ if q is contravariant.

(L2) If p is contravariant, then $q^*(p^*(A \wedge B) \wedge B) \vdash q^*(p^*(A))$ if q is covariant, and $q^*(p^*(A)) \vdash q^*(p^*(A \wedge B) \wedge B)$ if q is contravariant.

On the one hand, (L1) in conjunction with a contraction rule
(C) $q^*(B) \vdash q^*(B \wedge B)$ if q is covariant, and $q^*(B \wedge B) \vdash q^*(B)$ if q is contravariant, gives Peirce's iteration rule for covariant contexts:

$$p^*(A) \wedge B \vdash p^*(A) \wedge B \wedge B \vdash p^*(A \wedge B) \wedge B.$$

On the other hand, (L2) in conjunction with contraction gives Peirce's de-iteration rule for contravariant contexts:

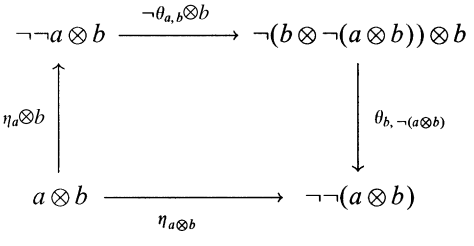
$$p^*(A \wedge B) \wedge B \vdash p^*(A \wedge B) \wedge B \wedge B \vdash p^*(A) \wedge B.$$

The de-iteration rule for covariant contexts, and iteration for contravariant contexts, are both instances of the weakening rule: we have, respectively,

$$p^*(A \wedge B) \wedge B \vdash p^*(A) \wedge B,$$
$$p^*(A) \wedge B \vdash p^*(A \wedge B) \wedge B.$$

Let us define the entailment relation for linear Alpha as the smallest reflexive and transitive relation on alpha graphs which includes the entailments $A \vdash B$ given by (L1), (L2), and double sep introduction/elimination. Then linear Alpha is equivalent to propositional linear logic (i.e., multiplicative linear logic without the storage and co-storage modalities ! and ?), in the sense that the class of sequents which are provable in linear Alpha is the same as the class of sequents provable in propositional linear logic. This result may be regarded as a linear analogue of our representation theorem, and is proved in essentially the same way. On a more sophisticated categorical level, this result corresponds to the following "alpha-style" definition of $*$-autonomous categories:

Definition 8. A $*$-autonomous category is a symmetric monoidal category $(V, \otimes)$ equipped with a strong contravariant self-adjoint equivalence $\neg : V \to V$ such that the following diagram commutes:



6. Typed Alpha

In earlier sections, we presented Alpha as a single-sorted or single-typed algebraic theory. With a view toward our forthcoming treatment of Peirce's system Beta, we

conclude this paper with a multi-typed version of Alpha. Since Alpha is equivalent to the theory of Boolean algebras, it is enough to present the framework in Boolean terms, but at the end we will translate the framework into the more geometric language of Alpha.

6.1. Languages and theories

In what follows, we let C_0 denote the class of objects of a category C, and S^* the underlying set of the free monoid on S.

Definition 9. A typed language consists of a set S and a function $\lambda : S^* \to \mathscr{S}et_0$, sending each word $w \in S^*$ to a set $\lambda(w)$.

Alternatively, a typed language is given by a set S and a function $\tau : P \to S^*$, where the fiber $\tau^{-1}(w)$ is the same as the set $\lambda(w)$. The intuition is that P is a set of formal predicates, and the function $\tau : P \to S^*$ is a typing function which sends a predicate p to its type $\tau(p) = (x_1, \ldots, x_n)$.

To give a set-theoretic model of a typed language, we need two pieces of data. The first interprets each element $x \in S$ as a set, say $M(x)$. The function M is extended to words $w \in S^*$ by interpreting $M(w)$ for $w = (x_1, \ldots, x_n)$ as a product:

$$M(w) = M(x_1) \times \cdots \times M(x_n).$$

The second datum interprets predicates p of type $w = (x_1, \ldots, x_n)$ as n-ary relations, i.e., as subsets of $M(w)$, or as elements of the power set $PM(w)$. We reformulate this notion of model in a way that invites easy generalization.

First, as a convenience, we blur the distinction between a set and the discrete category on that set, so that a typed language is a set S together with a functor $\lambda : S^* \to \mathscr{S}et$. The first datum of a set-theoretic model is then a monoidal functor $M : S^* \to \mathscr{S}et$ which sends products in S^* to cartesian products in $\mathscr{S}et$. The second datum, which sends predicates to relations, can be expressed as follows: let $P : \mathscr{S}et \to \mathscr{B}ool$ be the contravariant power set functor from sets to Boolean algebras. Let $U : \mathscr{B}ool \to \mathscr{S}et$ be the underlying set functor. Then the second datum is equivalent to a S^*-indexed collection of functions

$$\lambda(w) \to UPM(w),$$

that is to say, a natural transformation $\mu : \lambda \to UPM$.

More generally, let C be any category with (cartesian) products, and let $T : C \to \mathscr{B}ool$ be any contravariant functor. Such a pair (C, T) is called a *propositional theory*. It is straightforward to define a model of a typed language $(S, \lambda : S^* \to \mathscr{S}et)$ in a propositional theory (or propositional model, for short): it consists of a monoidal functor $M : S^* \to C$ together with a natural transformation $\mu : \lambda \to UTM$. One may regard a

propositional model as a translation from a typed language to the underlying typed language of a propositional theory, as we now explain.

Definition 10. A translation $(S,\lambda)\to(S',\lambda')$ between two typed languages consists of a function $f:S\to S'$ together with a natural transformation $\lambda\to\lambda' f^*$.

Definition 11. A translation $(C,T)\to(C',T')$ between two propositional theories consists of a product-preserving functor $F:C\to C'$ together with a natural transformation $\xi:T\to T'F$.

Defining the composition of translations is straightforward. For example, in the propositional theory case, if we have translations $(F,\xi):(C,T)\to(C',T')$ and $(F',\xi'):(C',T')\to(C'',T'')$, the composite translation is given by $F'F$ together with the composite transformation

$$T\to T'F\to T''F'F,$$

where the first map is ξ and the second is $\xi'F$. We thus obtain the category of typed languages $\mathscr{L}ang$ and the category of propositional theories $\mathscr{T}h$, where in each case the morphisms are translations.

Remark. Technically, one should speak of a 2-category of theories. If (F,ξ) and (G,θ) are both translations of the form $(C,T)\to(C',T')$, then a *modification* between them consists of a transformation $\eta:F\to G$ such that $(T'\eta)\cdot\xi=\theta$. Then theories, translations, and modifications form a 2-category.

There is an underlying functor from theories to languages, as follows. Given a theory (C,T), its underlying language is given by the set of objects $S=C_0$, together with a map λ given by the composite

$$C_0^*\xrightarrow{\pi}C\xrightarrow{T}\mathscr{B}ool\xrightarrow{U}\mathscr{S}et,$$

where π is a monoidal functor which extends the inclusion $i:C_0\to C$. (Many choices of π are possible, but all such monoidal functors are naturally isomorphic.) Having chosen a π for each theory (C,T) (and say π' for (C',T')), we have for each translation $(F,\xi):(C,T)\to(C',T')$ a canonical natural isomorphism

$$\alpha:F\pi\cong\pi'F_0^*$$

between monoidal functors of the form $C_0^*\to C'$. Then the underlying language translation of the theory translation (F,ξ) is given by the function

$$F_0:C_0\to C_0',$$

together with the transformation

$$(UT'\alpha)(U\xi\pi) : UT\pi \to UT'\pi' F_0^*.$$

This completes the description of the underlying functor $U : \mathscr{T}h \to \mathscr{L}ang$.

If $(S, \lambda : S^* \to \mathscr{S}et)$ is a typed language and $(C, T : C \to \mathscr{B}ool)$ is a propositional theory, then a translation $(S, \lambda) \to U(C, T)$ is precisely a propositional model of (S, λ) in (C, T), for such a translation is a function $f : S \to C_0$ together with a transformation

$$\mu : \lambda \to UT\pi f^*,$$

where πf^* is a monoidal functor $S^* \to C$ extending $f : S \to C$. We wish to solve the universal mapping problem of extending such a propositional model (f, μ) to a translation of propositional theories. That is to say, we are interested in constructing a left adjoint $F : \mathscr{L}ang \to \mathscr{T}h$ to $U : \mathscr{T}h \to \mathscr{L}ang$.

We call $F(S, \lambda)$ the free propositional (or Boolean, or Alpha) theory on the typed language (S, λ).

The construction of $F(S, \lambda)$ is straightforward. The first step is to construct the free cartesian category $\Pi(S)$ on the set S (the free "category with finite cartesian products" generated by S), so that the category of functors $f : S \to C$ is naturally equivalent to the category of product-preserving functors

$$\hat{f} : \Pi(S) \to C.$$

The construction of $\Pi(S)$ falls into a class of similar constructions known as categorical wreath products [9]; actually what we do is construct the free category with finite coproducts $\Sigma(S)$ and take its opposite. Let Fin (for "finite sets") be the category whose elements are natural numbers $n \geq 0$ and whose morphisms are functions $\{1, \ldots, m\} \to \{1, \ldots, n\}$. Fin is the free "category with finite coproducts" on a single generator. We construct $\Sigma(S)$ as a wreath product $Fin \int S$, i.e., the category whose objects are pairs

$$(m; x = \langle x_1, \ldots, x_m \rangle)$$

consisting of an object $m \in Fin_0$ and an m-tuple $\langle x_1, \ldots, x_m \rangle$ of elements of S (this should be regarded as a formal coproduct of the x_i), and whose morphisms $f : (m, x) \to (n, y)$ are maps $f : m \to n$ in Fin such that $x_i = y_{f(i)}$ for $1 \leq i \leq m$. There is an inclusion map $j : S \to Fin \int S$ which sends $x \in S$ to the object $(1; x)$; it is straightforward to verify that j is indeed (2-)universal among functors from S into categories with finite coproducts.

Remark. Given a category C equipped with a functor $F : C \to \mathscr{S}et$, and a category D, the wreath product $C \int D$ has for its objects pairs $(c, x : F(c) \to D)$ consisting of objects c in C and functors from the discrete category $F(c)$ to D. Morphisms $(c, x) \to (d, y)$

are pairs (f,θ), where $f:c\to d$ is a morphism in C and $\theta:x\to y\cdot F(f)$ is a natural transformation. As a special case, we have the classical wreath products in group theory, constructed from data (G,X,H), where G and H are groups and X is a set with an action by G. The classical wreath product $G\int H$ is the semi-direct product of G with the induced action of G on the group $\hom(X,H)$, the X-fold product of H. This construction $G\int H$ is a special case of the wreath products being described here, where G and H are regarded as one-object categories and the permutation representation of G on X is rendered as a functor $X:G\to\mathscr{S}et$.

To complete the construction of the free propositional theory on (S,λ), let us extend $j:S\to\Pi(S)$ to a monoidal functor $\sigma:S^*\to\Pi(S)$. Monoidal functors $S^*\to C$ are determined up to isomorphism by their restrictions to S; it follows that there is an isomorphism

$$(\pi f^*:S^*\to C_0^*\to C)\cong(\hat{f}\sigma:S^*\to\Pi(S)\to C).$$

Thus, the datum $\mu:\lambda\to UT\pi f^*$ of a propositional model corresponds bijectively to

$$\lambda\to UT\hat{f}\sigma.$$

Now the functor $-\circ\sigma$, which sends presheaves on $\Pi(S)$ to presheaves on S^*, has a left adjoint given by left Kan extension Lan_σ. This will be computed in a moment. The last map corresponds bijectively to

$$Lan_\sigma\lambda\to UT\hat{f}$$

and now if F denotes the free Boolean algebra functor $\mathscr{S}et\to\mathscr{B}ool$, this in turn corresponds bijectively to

$$FLan_\sigma\lambda\to T\hat{f}.$$

By this calculation, we have the following theorem.

Theorem 2. *$(\Pi(S),FLan_\sigma\lambda:\Pi(S)\to\mathscr{B}ool)$ is the free propositional theory on the typed language $(S,\lambda:S^*\to\mathscr{S}et)$.*

To calculate Lan_σ, observe that the functor $\sigma:S^*\to\Pi(S)$ is given by a bijection $\rho:S^*\to\Sigma(S)_0=\Pi(S)_0$. We also recall that $\Sigma(S)$ was given by a wreath construction $Fin\int S$. The functor Lan_σ, which is of the form

$$\mathscr{S}et^{S^*}\to\mathscr{S}et^{Fin\int S},$$

sends a functor $\lambda:S^*\to\mathscr{S}et$ to a functor whose value at $T\in(Fin\int S)_0$ is given by

$$(Lan_\sigma\lambda)(T)=\sum_{\rho(w)\to T}\lambda(w),$$

where the coproduct is taken over all morphisms of the form shown, with w ranging over S^*. The action of $Lan_\sigma\lambda$ on morphisms $T \to T'$ is given by re-indexing the coproducts.

The free Boolean algebra functor $F : \mathscr{S}et \to \mathscr{B}ool$ is well-known, but it will be convenient to formulate it as follows. If X is a set, we let X^- denote the restriction of the representable functor $\hom(-, X)$ to a contravariant functor $Fin \to \mathscr{S}et$. We also have a covariant functor $B_- : Fin \to \mathscr{S}et$, sending the finite set $\{1, \ldots, n\}$ to (the underlying set of) B_n, the free Boolean algebra on n generators. The underlying set of FX may then be expressed as a "tensor product" [15]

$$B_- \otimes_{Fin} X^-.$$

More explicitly, this is a quotient of the set

$$\sum_{n \geq 0} B_n \times X^n,$$

where one mods out by the equivalence relation $\sim$ generated by

$$(B_f(p), \vec{x}) \sim (p, X^f(\vec{x})),$$

where $f : m \to n$ ranges over morphisms of Fin, where $B_f : B_m \to B_n$ and X^f are the values of f under the functors B_- and X^-, where p ranges over B_m, and where $\vec{x}$ ranges over X^n.

Thus, the free propositional theory on (S, λ) consists of $\Pi(S)$ plus a covariant functor of the form

$$Fin \int S \to \mathscr{B}ool$$

whose value at an object $T = (m, \langle x_1, \ldots, x_m \rangle)$ is a certain quotient of the set

$$\sum_{n \geq 0} B_n \times (Lan_\sigma\lambda)(T)^n.$$

An element of this quotient set shall be called an alpha graph of type T.

6.2. Typed Alpha graphs

To get a clearer and more geometric representation of such typed alpha graphs, we first view an element of $(Lan_\sigma\lambda)(T)^n$ as an n-tuple of predicates p_i of type w_i, each indexed by an arrow of the form

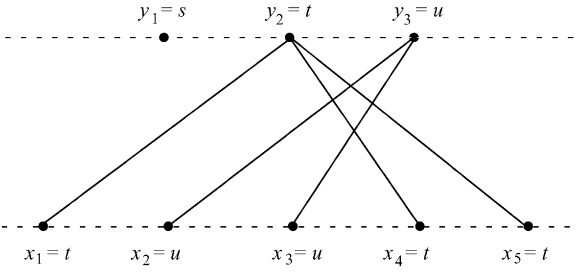
$$\rho(w_i) \to T.$$

Since these arrows live in $Fin\int S$, the free category-with-coproducts, they can be collected into a single map in $Fin\int S$:

$$\phi : \sum_{i=1}^{n} \rho(w_i) \to T.$$

A map in $Fin\int S$ will be depicted by taking the "directed graph" of its underlying function $f : \{1,\ldots,m\} \to \{1,\ldots,n\}$ in Fin, and labeling its connected components by elements in S. In more detail, the elements of the domain $\{1,\ldots,m\}$ will be depicted in left-to-right order as points on a horizontal line $y = a$ in the plane, and similarly the elements of the codomain as points on $y = b$, with $a < b$. For each pair $(i, f(i))$ where $i \in \{1,\ldots,m\}$, there is an edge with source i and target $f(i)$. Observe that each connected component of this directed graph corresponds bijectively to an element $j \in \{1,\ldots,n\}$. Each such component is then assigned the label y_j, if the given morphism in $Fin\int S$ is of the form $(m, x = \langle x_1,\ldots,x_m\rangle) \to (n, y = \langle y_1,\ldots,y_n\rangle)$. (Notice $x_i = y_j$ if i maps to j.)

Example.



An element of B_n can be represented by an alpha graph G whose nodes are labeled in $\{x_1,\ldots,x_n\}$. At the same time, an n-tuple of predicates $p_1,\ldots,p_n$ may be represented by a function

$$\{x_1,\ldots,x_n\} \to P$$

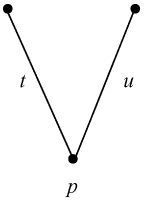
into the set of predicates P. Together, these two items may be assembled to yield an alpha graph with nodes labeled in $\{(x_1, p_1),\ldots,(x_n, p_n)\}$. However, in passing to the quotient $B_- \otimes_{Fin} P^-$, the information recorded in the x_i-components of these labels is lost, so that all that is needed to describe an element of this quotient is an alpha graph with nodes labeled in P.

Next, the type $w = \tau(p) \in S^*$ of a predicate p may be depicted by drawing "strings" in the plane which end at a node labeled p. If, for example, $w = (s_1,\ldots,s_n)$, then we draw n strings; up to affine change in each of the coordinates (x, y), the point p may be taken to be $(0, 0)$, and the ith string as a curve described parametrically by

$$t \mapsto (x_i(t), t),$$

where $t \in [0, 1]$, where $x_1(t) < \cdots < x_n(t)$ for $0 < t \leq 1$, and where $x_i(0) = 0$ for $1 \leq i \leq n$.

Example.

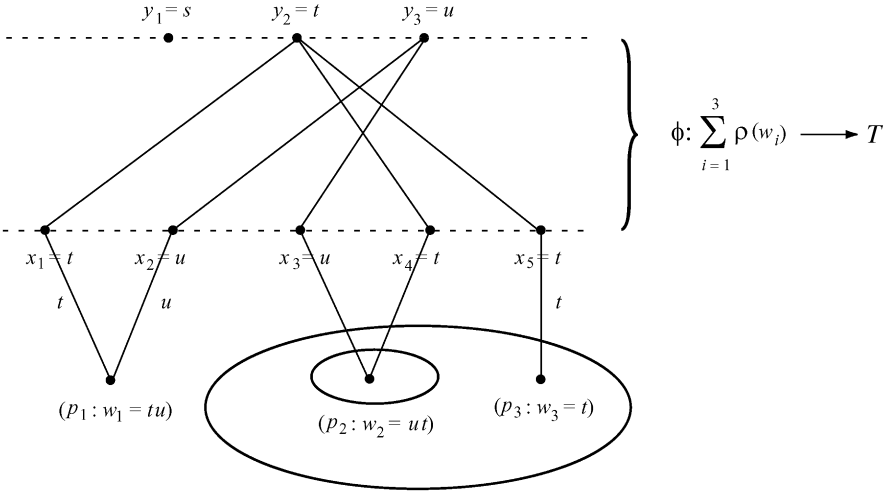


Finally, an alpha graph of type T, represented by the various data

$$G \in \alpha_n \qquad (p_1, \ldots, p_n) \in \lambda(w_1) \times \cdots \times \lambda(w_n) \qquad \phi : \sum_{i=1}^{n} \rho(w_i) \to T,$$

may be depicted by coherently pasting together the geometric pictures given above for each of these data.

Example. Here is an alpha graph of type $T = stu \in S^*(= \Pi(S)_0)$. Each predicate p_i lives in $\lambda(w_i)$; we indicate this by the notation $p_i : w_i$.



Acknowledgements

It is our pleasure to acknowledge the help of Stuart Kurtz, who with the first author initiated a mathematical study of existential graphs [11]. We are also grateful to Jens Erik Fenstad and Saunders Mac Lane for their encouragement and continued interest in this project.

References

[1] G. Allwein, J. Barwise, Logical Reasoning with Diagrams, Oxford University Press, Oxford, 1996.
[2] R.F. Blute, J.R.B. Cockett, R.A.G. Seely, T.H. Trimble, Natural deduction and coherence for weakly distributive categories, J. Pure Appl. Algebra 113 (1996) 229–296.
[3] J.Y. Girard, Linear logic, Theoret. Comput. Sci. 50 (1987) 1–102.
[4] J.Y. Girard, Y. Lafont, P. Taylor, Proofs and Types, Cambridge Tracts in Theoretical Computer Science, vol. 7, Cambridge University Press, Cambridge, 1989.
[5] E.M. Hammer, Logic and Visual Information, CSLI Publications, Stanford, 1996.
[6] N. Houser, Introduction to volume 4, in: Writings of Charles S. Peirce, Indiana University Press, Indianapolis, 1989.
[7] A. Joyal, R. Street, An introduction to Tannaka duality and quantum groups, in: A. Carboni, M.C. Pedicchio, G. Rosolini (Eds.), Category Theory, Proc. Internat. Conf., Como, Italy, Springer, Berlin, 1991, pp. 411–492.
[8] A. Joyal, R. Street, The geometry of tensor calculus I, Adv. Math. 88 (1991) 55–112.
[9] G.M. Kelly, On Clubs and Doctrines, Lecture Notes in Mathematics, vol. 420, Springer, Berlin, 1974, pp. 181–250.
[10] G.M. Kelly, Basic Concepts of Enriched Category Theory, London Math. Soc. Lecture Note Series, vol. 64, Cambridge University Press, Cambridge, 1982.
[11] S.A. Kurtz, G. Brady, Existential Graphs I: Alpha, <http://faith.cs.uchicago.edu/peirce/>, 1997.
[12] J. Lambek, Deductive Systems and Categories II, Lecture Notes in Mathematics, vol. 86, Springer, Berlin, 1969, pp. 76–122.
[13] F.W. Lawvere, Functorial semantics of algebraic theories, Proc. Natl. Acad. Sci. 50 (1963) 869–873.
[14] S. Mac Lane, Natural associativity and commutativity, Rice Univ. Studies 49 (1963) 28–46.
[15] S. Mac Lane, The Milgram Bar Construction as a Tensor Product of Functors, Lecture Notes in Mathematics, vol. 168, Springer, Berlin, 1970, pp. 135–152.
[16] S. Mac Lane, Categories for the Working Mathematician, Graduate Texts in Mathematics, vol. 5, Springer, Berlin, 1971.
[17] E.G. Manes, Algebraic Theories, Graduate Texts in Mathematics, vol. 26, Springer, Berlin, 1976.
[18] J.P. May, The Geometry of Iterated Loop Spaces, Lecture Notes in Mathematics, vol. 271, Springer, Berlin, 1972.
[19] C.S. Peirce, Prolegomena to an apology for pragmaticism, The Monist 16 (1906) 492–546. Reprinted in: C. Hartshorne and P. Weiss (Eds.), Collected Papers of Charles Sanders Peirce, vol. 4, Harvard University Press, Cambridge, 1933, pp. 293–410.
[20] D.D. Roberts, The existential graphs of Charles S. Peirce, Ph.D. Thesis, University of Illinois, 1963.
[21] D.D. Roberts, The Existential Graphs of Charles S. Peirce, Mouton, The Hague, 1973.
[22] R.A.G. Seely, Linear logic, $*$-autonomous categories, and cofree coalgebras, in: J. Gray and A. Scedrov (Eds.), Categories in Computer Science and Logic, Contemporary Mathematics, vol. 92, Am. Math. Soc., Providence, RI, 1989, pp. 371–382.
[23] J.F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA, 1984.
[24] J.J. Zeman, The graphical logic of C.S. Peirce, Ph.D. Thesis, University of Chicago, 1964.