South American Journal of Logic
Vol. 1, n. 2, pp. 401–434, 2015
ISSN: 2446-6719

Conditional Reasoning with String Diagrams

Clayton Peterson

Abstract

String diagrams are useful tools in many disciplines, including physics, computer science and linguistics. Their language is closely related to category theory. From an epistemological point of view, category theory offers a foundational framework that enables us to see the common structure shared by these disciplines, namely the structure of a monoidal category. This structure can also be found in other fields of study, for instance in logic. Accordingly, it appears that string diagrams could be applied in logic to model reasonings. In this research article, we apply string diagrams to deontic logic and use them to model conditional normative inferences. Building on previous work, we define a deductive system for conditional normative reasoning $\mathcal{CNR}$ and show that it is sound and complete with respect to a string diagrammatic semantics. A decision procedure is provided and string diagrams are used to test the validity of conditional inferences.

Keywords: Deductive system, Deontic logic, Normative inference, Validity

1 Introduction

String diagrams are useful tools in physics and computer science. They allow to visually represent the flow of information and the transformation processes. Recently, Baez and Stay [3] proposed a string diagram calculus and showed how it offers common grounds not only for physics and computer science, but also for logic and topology. Their proposition is twofold. First, they argue that category theory offers a rich and powerful foundational framework that enables us to abstract the common structure shared by different fields of study. In this respect, they show how it offers a common framework for physics, computer science, topology and logic. Secondly, they insist on the fact that string diagrams offer an alternative way to present categorical notions (on this point, see also [30]). Given that string diagrams are closely related to the language of category theory, Baez and Stay argue that string diagrams could be used as a foundational

framework, allowing a visual representation of the transformation processes that are modeled within these disciplines.

In logic, these transformation processes happen between two formulas through a consequence relation. Representing a *proof* or a *deduction* by an arrow $\varphi \longrightarrow \psi$, a reasoning can be viewed as a transformation process where we input a list of formulas (the conjunction of the premises) and obtain a new one (the conclusion) through the application of some rules and axioms. From a categorical standpoint, logical systems can be proven to be instances of free categories where arrows are proofs and objects are formulas. Historically, this categorical understanding of logic is due to the groundbreaking work of Lambek [19, 20][1], who provided a foundational framework that enables us to classify different deductive systems as (free) categories with specific structures.[2] In light of Baez's and Stay's proposal, this framework allows us to construct a string diagrammatic semantics for deductive systems. Using diagrams as a semantics is not in itself something new. For instance, Girard [12] used a proof net semantics to model linear logic. The interest of Baez's and Stay's approach is that their string diagrammatic language can be applied to a wide variety of deductive systems and substructural logics.

The aim of the present paper is to apply Baez's and Stay's string diagrams to our previous work in deontic logic and normative reasoning (cf. [26, 27]). The objective is to introduce a diagrammatic semantics that allows to visually represent and to test the validity of conditional normative inferences. To make this paper self-sufficient, we begin by summarizing the foundational framework we adopt in section 2, where we expose the relationship between category theory, monoidal deductive systems and string diagrams. Then, we briefly present in section 3 the deductive system $\mathcal{CNR}$. This logic was introduced in [27] and is meant to model conditional normative reasoning. We show that it is sound and complete with regards to a string diagrammatic semantics and, further, we use string diagrams to show that it is decidable. In light of the decidability of $\mathcal{CNR}$, we analyze Chisholm's [8] paradox in section 4 to exemplify how string diagrams can be used to test the validity of normative inferences. We conclude in section 5 with avenues for future research.

[1] See also Lambek and Scott [21].

[2] Note that we only partially follow Lambek in his categorical understanding of logic. We do not define deductive systems as categories but rather use their categorical properties to classify them. See [28, 27] for details.

2 Category theory, string diagrams and deductive systems

String diagrams can be used in physics (e.g., [3]) and computer science (e.g., [23]) to represent the flow of information and the transformation processes. They can also be used in linguistic to model meaning (cf. [9]). This is possible given that these structures can be compared on the basis that they all involve instances of *monoidal categories*, which are coherent with specific diagrammatic languages (cf. [30]).

Following Mac Lane [22], a *monoidal category* is a category $\mathbb{C}$ equipped with an associative tensor product $\otimes$ (a functor) and a unit object 1, satisfying the following natural isomorphisms together with the triangle and pentagon identities.[3]

$$a_{\varphi,\psi,\rho} : (\varphi \otimes \psi) \otimes \rho \longrightarrow \varphi \otimes (\psi \otimes \rho)$$
$$l_{\varphi} : 1 \otimes \varphi \longrightarrow \varphi$$
$$r_{\varphi} : \varphi \otimes 1 \longrightarrow \varphi$$

In a nutshell, a category is a structure where each object has an identity arrow that satisfies the identity laws (i.e., given $f : \varphi \longrightarrow \psi$ and $g : \psi \longrightarrow \rho$, $f = 1_{\psi} f$ and $g = g1_{\psi}$) and where composition $gf : \varphi \longrightarrow \rho$ for two arrows $f : \varphi \longrightarrow \psi$ and $g : \psi \longrightarrow \rho$ is defined and is associative (i.e., given $h : \rho \longrightarrow \tau$, $(hg)f = h(gf)$). The category becomes monoidal when we add an associative operation that comes with an object that satisfies the aforementioned equations.

A *symmetric* monoidal category comes with a braiding morphism (a natural transformation) $\beta_{\varphi,\psi} : \varphi \otimes \psi \longrightarrow \psi \otimes \varphi$. This morphism is its own inverse.[4] In a symmetric monoidal category, the tensor product is commutative and it respects the hexagon identities.

A monoidal category is *closed* when the tensor product possesses a right adjoint functor $\multimap$, which happens when $\mathrm{Hom}_{\mathbb{C}}(\varphi \otimes \psi, \rho) \cong \mathrm{Hom}_{\mathbb{C}}(\varphi, \psi \multimap \rho)$.[5]

Following [28, 27], consider a language $\mathcal{L} = \{Prop, (,), \otimes, 1, \multimap, \triangleright, 0\}$, where $Prop$ is a collection of atomic formulas p_i and well-formed formulas are recursively defined as follows:

$$\varphi := p_i \mid 0 \mid 1 \mid \varphi \otimes \psi \mid \varphi \multimap \psi \mid \varphi \triangleright \psi$$

In this language, $\otimes$ is a tensor with unit 1 and $\multimap$ /$\triangleright$ are the conditionals (i.e., its ajdoints). Monoidal logics are defined using the rules and axiom schemata presented in figure 1. Negations are defined by $\sim \varphi =_{df} \varphi \multimap 0$ and $\neg\varphi =_{df} \varphi \triangleright 0$. A double line means that the rule can be applied both top-down and bottom-up.

[3] See [22] for the explicit categorical definitions.

[4] If not, then it is a *braided* category.

[5] Note that if the category is not symmetric, then there can be two different adjoints $\multimap$ and $\triangleright$ such that $\mathrm{Hom}_{\mathbb{C}}(\varphi \otimes \psi, \rho) \cong \mathrm{Hom}_{\mathbb{C}}(\psi, \varphi \triangleright \rho)$.

$$\frac{}{\varphi \longrightarrow \varphi}\ (1) \qquad \frac{}{\sim \neg\varphi \longrightarrow \varphi}\ (\sim \neg) \qquad \frac{}{\neg \sim \varphi \longrightarrow \varphi}\ (\neg \sim) \qquad \frac{\varphi \otimes \psi \longrightarrow \rho}{\psi \longrightarrow \varphi \triangleright \rho}\ (\text{cl'})$$

$$\frac{\varphi \longrightarrow \psi \qquad \psi \longrightarrow \rho}{\varphi \longrightarrow \rho}\ (\text{cut}) \qquad \frac{\varphi \longrightarrow \psi \qquad \rho \longrightarrow \tau}{\varphi \otimes \rho \longrightarrow \psi \otimes \tau}\ (\text{t}) \qquad \frac{\varphi \otimes \psi \longrightarrow \rho}{\varphi \longrightarrow \psi \multimap \rho}\ (\text{cl})$$

$$\frac{\varphi \longrightarrow \psi \otimes 1}{\varphi \longrightarrow \psi}\ (\text{r}) \qquad \frac{\varphi \longrightarrow 1 \otimes \psi}{\varphi \longrightarrow \psi}\ (\text{l}) \qquad \frac{\tau \longrightarrow (\varphi \otimes \psi) \otimes \rho}{\tau \longrightarrow \varphi \otimes (\psi \otimes \rho)}\ (\text{a}) \qquad \frac{\varphi \longrightarrow \psi \otimes \tau}{\varphi \longrightarrow \tau \otimes \psi}\ (\text{b})$$

$$\frac{\varphi \longrightarrow \psi \qquad \varphi \longrightarrow \rho}{\varphi \longrightarrow \psi \otimes \rho}\ (\otimes\text{-in}) \qquad \frac{\varphi \longrightarrow \psi \otimes \rho}{\varphi \longrightarrow \psi}\ (\otimes\text{-out}) \qquad \frac{\varphi \longrightarrow \psi \otimes \rho}{\varphi \longrightarrow \rho}\ (\otimes\text{-out})$$

Figure 1: Rules and axiom schemata for monoidal logics

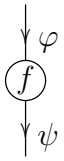
A *deductive system* is defined as a collection of formulas together with a collection of equivalence classes of proofs satisfying an axiom (1) expressing the reflexivity of the consequence relation and a rule (cut) representing its transitivity. From a categorical perspective, a deductive system can be proven to be an instance of a free category where objects are formulas and arrows are (equivalence classes of) proofs. Depending on the rules and axiom schema that are adopted, one can defined a *monoidal deductive system* as a deductive system satisfying (t), (a), (l) and (r), a *monoidal closed deductive system* as a monoidal deductive system satisfying (cl) and (cl'), a *symmetric deductive system* as a monoidal deductive system satisfying (b) or a *Cartesian deductive system* as a monoidal deductive system satisfying ($\otimes$-in) and ($\otimes$-out). Closed deductive systems can have classical negations through the satisfaction of ($\neg \sim$) and ($\sim \neg$). These deductive systems can be proven to be instances of free monoidal, closed, symmetric and Cartesian categories, respectively. The logical connectives can be proven to be functors that act on both formulas (objects) and proofs (arrows). Accordingly, deductive systems can be classified with respect to their categorical structure and their functorial properties.[6]

The interest of using category theory as a foundation for logic is that it enables us to bring to light the monoidal structure of logical systems. Considering that monoidal categories have specific string diagrammatic languages, it follows that we can define a string diagrammatic semantics for monoidal deductive systems. This semantics is not a mere artifact but is a powerful tool that can be used to determine the validity of normative inferences. There is a wide variety of monoidal categories and they all can be represented via a diagrammatic language. We refer the reader to the paper of Selinger [30] for a thorough presentation of the subject and to Joyal and Street [17, 16] for a more formal treatment. In what follows, we summarize the string calculus of Baez and Stay [3] and adapt it to our needs.

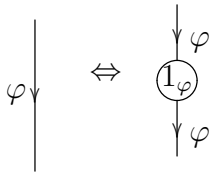
When one looks at logic from a categorical perspective, formulas are seen as objects

[6]If the deductive system is symmetric, $\multimap$ / $\sim$ can be reduced to $\triangleright$/$\neg$ and vice versa.

while proofs (deductions) are considered as arrows. In a string diagram language, formulas are considered as oriented strings while proofs are seen as transformation processes from one string to another. Hence, in string diagrams, a proof $f : \varphi \longrightarrow \psi$ is represented as follows. The orientation of the string is indicated by the arrowhead. In this case, f transforms the input φ into the output ψ.

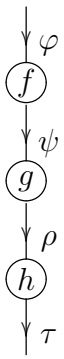


The *identity process* 1_φ is a transformation process that makes no change on a string φ.



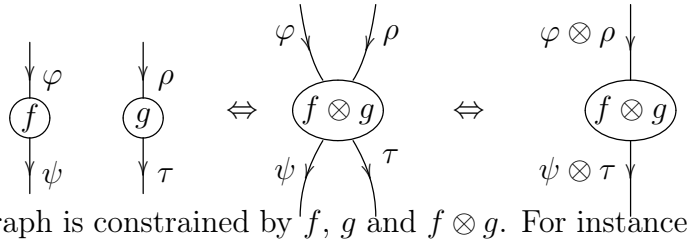
Equivalence between diagrams (denoted by '⇔') should be understood as a rewrite operation. Two diagrams are considered as 'equivalent' when one can be rewritten into the other, and vice versa (i.e., the graphs can be deduced from one another through the rewrite rules). For instance, the graph representing the string φ can be rewritten into the graph representing the identity process 1_φ that takes a string as an input, makes no change and then gives the string back as an output. Similarly, the identity process can be rewritten into the string φ. In this respect, these two graphs are equivalent. It should be noted, however, that equivalent graphs do not necessarily represent the same transformation processes (e.g., with the currying and uncurrying operations below).

This representation provides a language for categories, where the objects are strings and the arrows are transformation processes. It is easy to show that this language satisfies the definition of a category. Indeed, two strings $f : \varphi \longrightarrow \psi$ and $g : \psi \longrightarrow \rho$ can always be composed. Further, given $h : \rho \longrightarrow \tau$, the diagram of $h(gf)$ is equivalent to the diagram $(hg)f$. Indeed, we simply have to connect the strings to obtain $hgf : \varphi \longrightarrow \tau$.

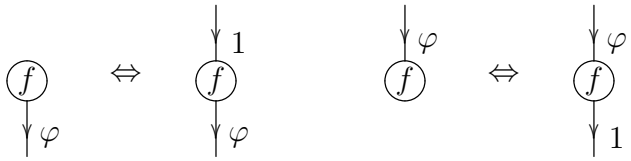


Moreover, composing with identity respects the identity laws (i.e., $1_\psi f = f$ and $g1_\psi = g$). In other words, one can strengthen at will either the string of the input or the string of the output.

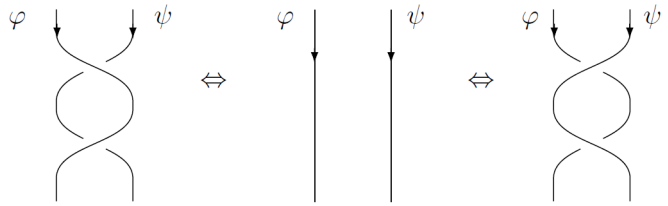
The string diagrammatic language for monoidal categories is obtained by requiring that composition of strings respects the representation of parallel processes.



Note that this graph is constrained by f, g and $f \otimes g$. For instance, if one replaced $f \otimes g$ by h in the aforementioned diagram, then one could not conclude legitimately that there are $f : \varphi \longrightarrow \psi$ and $g : \rho \longrightarrow \tau$. Given two processes f and g, one can make them run in parallel via $f \otimes g$.[7] The unit is drawn as a blank space:



The commutativity of the tensor product is obtained by allowing for braiding between strings. From this, we obtain that if we braid and then undo the braid, we get what we started with.

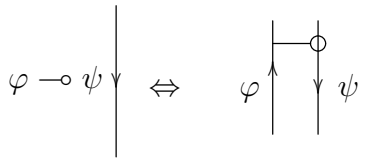


The symmetry of the braiding is obtained by requiring that the braiding rule is its own inverse. This is represented by the following rewrite rule:

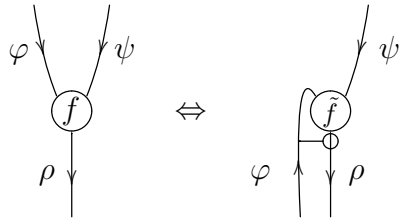
The representation of a conditional requires the introduction of an artifice (cf. [3]),

[7]This is related to the fact that $\otimes$ is a functor that acts on both objects and arrows. The arrow $f \otimes g : \varphi \otimes \rho \longrightarrow \psi \otimes \tau$ is given by $\otimes(f, g)$ form $f : \varphi \longrightarrow \psi$ and $g : \rho \longrightarrow \tau$.

namely a string oriented in the other direction.[8] Considering the orientation of a string, we can flip it over and obtain its *dual* string, going in the opposite direction. The little 'clasp' is used to bind the two strings together and redirect the orientation of the string φ.



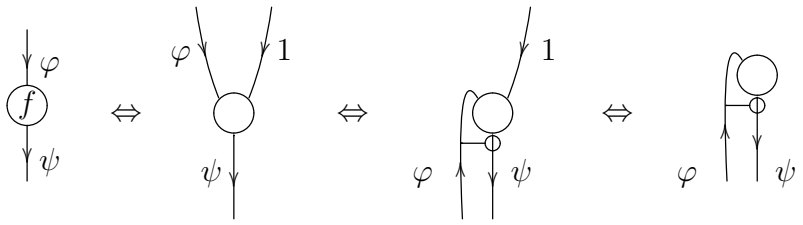
This 'claps' can be understood as a transformation process. The notation will be better understood if we consider the rewrite rules governing the relation between the tensor product and its adjoint. In string diagrams, the relation between $\otimes$ and $\multimap$ is represented by a currying and an uncurrying operation. The currying process is from left to right while uncurrying is from right to left.



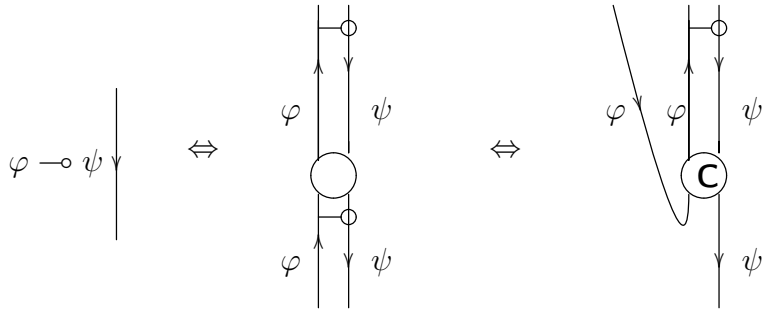
A string can be considered as an input when it is oriented towards a transformation process and, similarly, it can be seen as an output when it goes in the opposite direction. Here, the 'clasp' binds the two strings together and redirect the orientation of the string φ, turning $\varphi \multimap \rho$ into an output. Informally, currying means that one can bend down the string of the input φ and attach it to ρ to create an output $\varphi \multimap \rho$. In this case, the string φ is linked to the output ρ via the 'clasp' binding the strings together. Similarly, uncurrying means that one can 'unclasp' the strings and unbend the string φ. By doing so, the string φ becomes an input which is oriented towards the transformation process.

From this rewrite rule and the fact that the unit can be drawn as a blank space, we obtain that a string representing a conditional can be represented as a transformation process, and vice versa. That is, if we assume that there is a transformation process $f : \varphi \longrightarrow \psi$, then $\varphi \multimap \psi$ is the output of the unit and, similarly, if from the unit one gets as an output the conditional $\varphi \multimap \psi$, then there is a transformation process $f : \varphi \longrightarrow \psi$.

[8]Baez and Stay use dual strings to represent compact closed categories. Here, we only use it as an artifice to represent the conditionals. Hence, we will omit all the considerations regarding the zig-zag equations and the bubbles restricting their use.

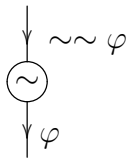


Hence, the notation is meant to suggest that a conditional is, to some extent, a transformation process. This can be exemplified further if we consider Ⓒ below. From the currying and uncurrying rewrite rules, we obtain the following equivalent diagrams.

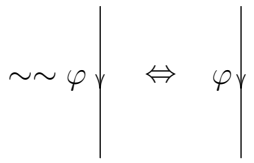


The string $\varphi \multimap \psi$ can be understood as a transformation process in light of Ⓒ. Indeed, if we input φ into Ⓒ, then the process returns φ upward, which is then redirected and transformed into ψ through the 'claps' binding the strings together and returned into Ⓒ to output ψ.[9] Therefore, the graph of the string $\varphi \multimap \psi$ is equivalent to the graph of a transformation process that allows to transform φ into ψ.

So far, we have presented the string diagrammatic language for a symmetric closed category. To suit our purpose, we need to augment Baez's and Stay's string diagrams with a transformation process allowing to eliminate double negations. Thus, we augment the language with a transformation process Ⓝ that erases the double negation.



From this, we obtain the following equivalence.



[9]This is an analogue to *modus ponens* in logic: from φ and $\varphi \multimap \psi$ one can derive ψ.

As we will see, we will add two other transformation processes to our string diagrammatic language. But first, let us present the logical system for conditional normative reasoning.

3 Logic for conditional normative reasoning

3.1 Syntax

We showed in [27] how defining a deontic logic as a symmetric closed deductive system with classical negation (which can be proven to be an instance of a free symmetric closed category) enables us to answer the arguments in favor of non-monotonic or adaptive foundations for deontic logic. These results were extended in [26] where we constructed a deontic deductive system to model conditional ($\mathcal{CNR}$) and unconditional ($\mathcal{OL}$) normative reasoning. Following [28], we defined these fragments as fibrations based on an action logic $\mathcal{AL}$ and a propositional action logic $\mathcal{PAL}$ and we presented a semantics within the framework of residuated partially ordered monoids (cf. [10]).

The aim of the present paper is to extend these results and provide a diagrammatic semantics for $\mathcal{CNR}$. This will be helpful to test the validity of conditional normative inferences. The logic for conditional normative reasoning $\mathcal{CNR}$ was initially defined on the grounds of an action logic $\mathcal{AL}$. For the purpose of the present article, we leave aside the semantical considerations of $\mathcal{AL}$ and concentrate only on $\mathcal{CNR}$. Nonetheless, we summarize the language of $\mathcal{AL}$ so the reader might understand the formulas in the scope of the deontic operators.

Given Act a collection of atomic actions a_i, the well-formed formulas $WFF_{\mathcal{AL}}$ (of type act, for actions) are recursively defined from the language $\mathcal{L}_{\mathcal{AL}} = \{Act, \bullet, *, \ominus, \curvearrowright, (,)\}$ as follows:

$$\alpha := \ a_i \mid \ * \mid \alpha \bullet \beta \mid \alpha \ominus \beta \mid \alpha \curvearrowright \beta$$

Here, $\bullet$ and $\curvearrowright$ are two different tensors with unit $*$. The connective $\ominus$ is the adjoint of $\bullet$. Briefly, $\bullet$ represents (simultaneous) action conjunction, $\curvearrowright$ action sequence and $\ominus$ means 'without'. Action negation is defined by $\alpha^* =_{df} * \ominus \alpha$ and $\mathcal{AL}$ is defined as a compact closed deductive system (see [28] for details).[10] It can be proven to be an instance of a free compact closed category (cf. [18]). The fragment $\{Act, \bullet, *, \ominus, \}$ is defined as a symmetric closed deductive system with classical negation satisfying (cpt1) and (cpt2) while the fragment $\{Act, *, \curvearrowright\}$ is defined as a monoidal deductive system. The connectives of $\mathcal{AL}$ are of the type that takes two formulas of type act and transforms them into another one. The action $*$ is both trivial (it makes no change) and impossible to perform.

[10] In this case, the notation for the adjoint is reversed. For instance, in comparison with the rules of figure 1, $* \ominus \alpha$ would be written $\alpha \multimap *$.

$$\frac{}{\alpha^* \bullet \beta \longrightarrow \beta \ominus \alpha}\ (\text{cpt2}) \qquad \frac{}{\beta \ominus \alpha \longrightarrow \alpha^* \bullet \beta}\ (\text{cpt1})$$

Now, consider a collection $Prop$ of atomic descriptive (declarative) formulas p_i of type d. Let AP be a collection of action propositions (of type ap) defined by the following condition: if $\alpha \in WFF_{\mathcal{AL}}$, then $\boldsymbol{\alpha} \in \mathsf{AP}$. The bold notation $\boldsymbol{\alpha}$ is used to represent the declarative sentence referring to the action α (see [28]). Assuming a well-formed formula α of type act, well-formed formulas of type np (for normative propositions) are recursively defined from the language $\mathcal{L}_{\mathcal{CNR}} = \{(,), \mathsf{AP}, Prop, \otimes, 1, \multimap, 0, O, P_s\}$ as follows:

$$\varphi := 1 \mid 0 \mid \boldsymbol{\alpha} \mid p_i \mid O\alpha \mid P_s\alpha \mid \varphi \otimes \psi \mid \varphi \multimap \psi$$

The connectives of $\mathcal{CNR}$ are of the type that takes either a proposition of type d and one of type np, a proposition of type ap and one of type np or two propositions of type np and transforms them into another proposition of type np. The logic $\mathcal{CNR}$ is defined as a symmetric closed deductive system with classical negation satisfying ($\mathbf{D}$) and ($\mathbf{P}$) with $\sim \varphi =_{df} \varphi \multimap 0$.

$$\frac{}{O\alpha \longrightarrow P_s\alpha}\ (\mathbf{D}) \qquad \frac{}{P_s\alpha \longrightarrow \sim O\alpha^*}\ (\mathbf{P})$$

The axiom schema ($\mathbf{D}$) states that what is obligatory is strongly permitted, while ($\mathbf{P}$) says that what is strongly permitted is weakly permitted. The operators O and P_s are of the type that takes an action and transforms it into a proposition of type np. Weak permission is defined by $P_w\alpha =_{df} \sim O\alpha^*$ and interdiction is defined by $F\alpha =_{df} O\alpha^*$.

$\mathcal{CNR}$ is defined as a symmetric closed deductive with classical negation. It can be proven to be an instance of a free symmetric closed category. If, further, it incorporates a co-tensor defined by $\varphi \oplus \psi =_{df} \sim \varphi \multimap \psi$, then it can be proven to be an instance of a free $*$-autonomous category (in the sense of [4]). To interpret $\mathcal{CNR}$ within framework of string diagrams, we need to upgrade the diagrammatic language with two special string processes Ⓓ and Ⓟ.

3.2 Completeness: prerequisites

Before going further, let us first introduce some terminology and notational conventions that will be used throughout the following constructions. A continuous string is composed of string(s) and/or transformation process(es).

Definition 3.1 (Path) *A* path *between two formulas φ and ψ is a continuous string.*

As we will see, the notion of *path* will be relevant for the definition of weak-inconsistency. To clearly understand the notion of a *path*, consider how we proceed to draw string diagrams, as illustrated in figure 2. A string diagram is drawn on a board $\mathcal{B}$ from different strings and transformation processes. Informally, consider a *toolbox* containing every string and transformation process schemata (obtained through the rewrite rules) for the string diagrammatic language of a symmetric closed category upgraded with $\textcircled{\sim}$, $\textcircled{\text{D}}$ and $\textcircled{\text{P}}$.

A string diagram is drawn on a board $\mathcal{B}$ from the tools in the toolbox and the strings that are assumed as hypotheses. As a notational convention, we will write $\mathcal{B} : \varphi_1, \ldots, \varphi_n$ to refer to a board where the formulas $\varphi_1, \ldots, \varphi_n$ are assumed as hypotheses. If the list is empty, we simply write $\mathcal{B}$. To draw a diagram, one can, at will, add to or remove from the board any string or transformation process from the toolbox. Note that the strings and transformation processes of the toolbox are reusable. However, the strings that are assumed as hypotheses cannot be removed from the board at will. If one starts a drawing on a board with some strings in the box of hypotheses, then one must end the drawing with these strings either on the board or in the hypotheses box. Note that adding an identity string from the toolbox to the board amounts to assume another hypothesis.

Having this terminology at our disposal, note that, as an example, there is no path from φ to ψ on $\mathcal{B}$ or $\mathcal{B} : \varphi, \psi$, but there is a path from $\varphi \otimes (\varphi \multimap \psi)$ to ψ.

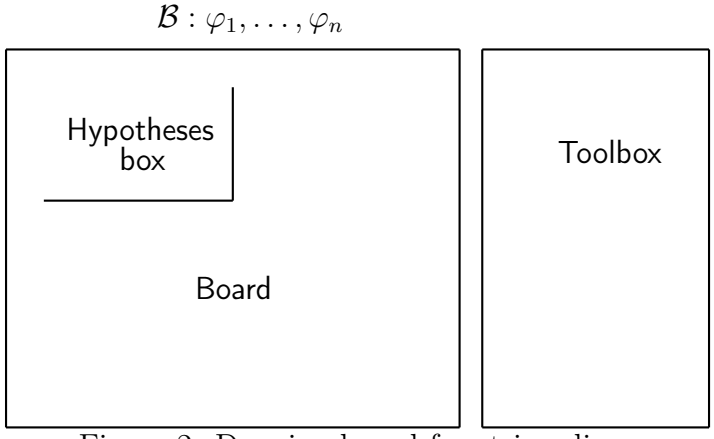


Figure 2: Drawing board for string diagrams

3.3 Soundness and completeness

We now turn our attention to the soundness and completeness theorems, or, in Selinger's [30] terms, the *coherence* theorem. It states that f is derivable within $\mathcal{CNR}$ if and only if it can be drawn as a string diagram.

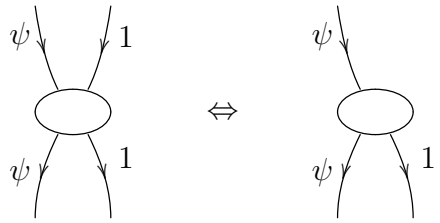
Consider a string diagrammatic language for a symmetric closed category upgraded with $\circledsim$, Ⓓ and Ⓟ.

Definition 3.2 (Validity) *A proof $f : \varphi \longrightarrow \psi$ is* valid *if and only if a path from φ to ψ can be drawn on every board $\mathcal{B} : \varphi_1, \ldots, \varphi_n$.*

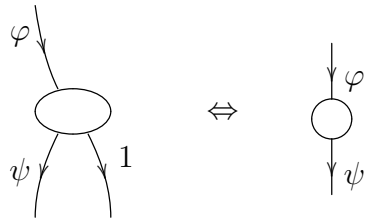
An equivalent formulation of this definition would be to say that f is valid if it can be drawn as a continuous string on $\mathcal{B}$. Indeed, f can be drawn on $\mathcal{B}$ if and only if it can be drawn on every $\mathcal{B} : \varphi_1, \ldots, \varphi_n$.

Theorem 3.3 *If $f : \varphi \longrightarrow \psi$ is a $\mathcal{CNR}$-arrow, then it is valid.*

The soundness theorem follows directly from the construction of the string diagrammatic language. Indeed, we only need to show that the rewrite rules preserve the validity of the rules and axioms of $\mathcal{CNR}$. In this respect, (1), (cut), (t), (a), (b), (cl), ($\sim\sim$), (**D**) and (**P**) are quite straightforward (their proof are left to the reader). The only exception is perhaps the rule (r). Bottom-up, it suffices to note that:



Hence, assuming $\varphi \longrightarrow \psi$, the rest follows from composition. Top-down, it suffices to draw 1 as a blank space.



Consequently, $\mathcal{CNR}$ is sound with respect to our string diagrammatic language.

The completeness theorem follows from the procedure that enables us to convert a string diagram into a proof. First, note that each tool within the toolbox has a proof-theoretical counterpart. The identity, double negation, obligation and permission processes are related to their corresponding axiom in $\mathcal{CNR}$. The diagrams for currying and uncurrying corresponds to (cl), the braiding diagram to (b) and the diagram for parallel processes to (t). Finally, composition of strings corresponds to (cut).

From the definition of validity, we know that $f : \varphi \longrightarrow \psi$ is valid if and only if there is a path from φ to ψ on every board $\mathcal{B} : \varphi_1, \ldots, \varphi_n$, which happens if and only if f can be drawn as a continuous string on $\mathcal{B}$.

The length l of a string diagram is determined by the number of strings and transformation processes required for its construction. Note that string diagrams are not necessarily finite.

Theorem 3.4 (Completeness) *If $f : \varphi \longrightarrow \psi$ can be drawn as a continuous string on $\mathcal{B}$, then f is a $\mathcal{CNR}$-arrow.*

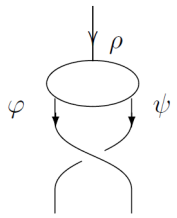
Proof. We proceed by induction on the length l of the diagram. If $l = 1$, then the string is either identity, double negation $\circledsim$, Ⓓ or Ⓟ. In this case, the diagram can be converted into a proof via their corresponding axiom in $\mathcal{CNR}$.

Now, assume that the property holds for $l = n$ but that it does not for $l = n + 1$. Consider the number of continuous strings on the board.

1. If there is only one continuous string on the board, then there are three possible combinations to obtain a diagram of length $n + 1$.

   (a) The diagram is obtained by adding another string on the board, in which case it is either the identity, double negation $\circledsim$, Ⓓ or Ⓟ.

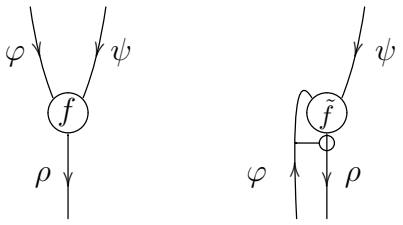
   (b) The string diagram of length $n + 1$ is obtained by an application of the braiding rule, in which case the diagram has the following form.



   By inductive hypothesis, we have a proof of $\rho \longrightarrow \varphi \otimes \psi$, and this can be converted into a proof via the braiding rule (b).

$$\dfrac{\dfrac{\vdots}{\rho \longrightarrow \varphi \otimes \psi}}{\rho \longrightarrow \psi \otimes \varphi}\,(\mathrm{b})$$

   (c) The string diagram of length $n+1$ is obtained by an application of currying or uncurrying. Hence, the string diagram of length n has either of one of these two forms.



By inductive hypothesis we have a proof of f or $\tilde{f}$, hence it can be converted into a proof by applying (cl).

$$\dfrac{\dfrac{\vdots}{\varphi \otimes \psi \longrightarrow \rho}}{\varphi \longrightarrow \psi \multimap \rho}\,(\mathrm{cl}) \qquad \dfrac{\dfrac{\vdots}{\psi \longrightarrow \varphi \multimap \rho}}{\varphi \otimes \psi \longrightarrow \rho}\,(\mathrm{cl})$$

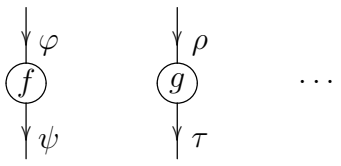
2. The diagram of length n contains a number $m > 1$ of continuous strings. By the inductive hypothesis, theses strings can be converted into proofs. The diagram of length $n + 1$ can be obtained in various ways.

   (a) The diagram is obtained by adding another string on the board, in which case it is either the identity, double negation ⓦ, Ⓓ or Ⓟ (and these strings can be converted into proofs).

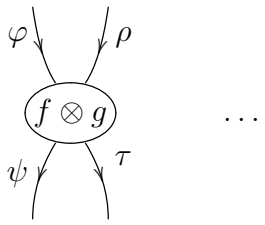
   (b) The diagram of length $n + 1$ is obtained by an application of the braiding rule, in which case 1(b) applies.

   (c) The diagram of length $n + 1$ is obtained by an application of currying or uncurrying, in which case 1(c) applies.

   (d) The diagram of length $n + 1$ is obtained by an application of the parallel process to two of its strings. In this case, the diagram of length n has the form:



   And the diagram of length $n + 1$ has the form:

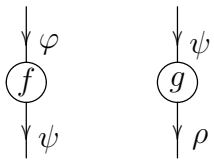


$\cdots$

This can be converted into a proof via the rule (t):

$$\dfrac{\dfrac{\vdots}{\varphi \longrightarrow \psi} \quad \dfrac{\vdots}{\rho \longrightarrow \tau}}{\varphi \otimes \rho \longrightarrow \psi \otimes \tau}\,(\mathrm{t})$$

(e) The diagram of length $n+1$ is obtained by an application of the composition process. Hence, we have two strings that can compose. This is represented by the following situation.



$\cdots$

In this case, a proof can be obtained through the application of the (cut) rule.

$$\dfrac{\dfrac{\vdots}{\varphi \longrightarrow \psi} \quad \dfrac{\vdots}{\psi \longrightarrow \rho}}{\varphi \longrightarrow \rho}\,(\mathrm{cut})$$

■

Let us now illustrate how the procedure works. Figure 3 shows how to convert the string diagram of $(p \multimap O\alpha) \otimes \sim\sim p \longrightarrow P_s\alpha$ into a proof.

We can see that the whole string is obtained by composing a braid with the string below point **1**. This substring is obtained by composing the string above point **2** with the transformation process Ⓓ. The string above **2** is obtained by composing a parallel process between two strings at point **4** (the one on the left resulting from composition with the double negation process at point **5**) with the uncurrying process at point **3**. This yields the following proof schema:

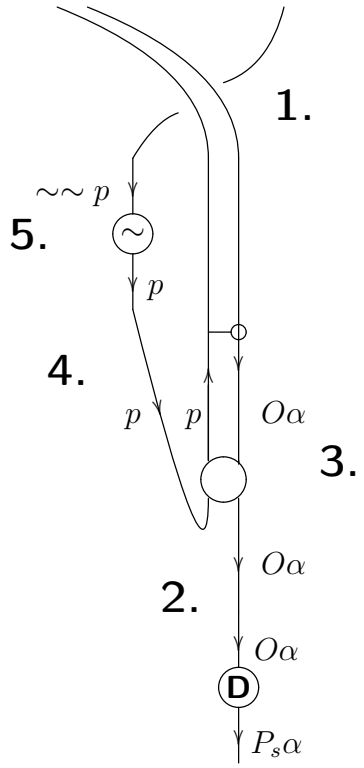


Figure 3: How to convert a diagram into a proof

$$\dfrac{\mathbf{5}\ \dfrac{}{\sim\sim p \longrightarrow p}\ (\sim\sim) \qquad \dfrac{}{p \multimap O\alpha \longrightarrow p \multimap O\alpha}\ (1)}{\mathbf{4}\ \dfrac{\sim\sim p \otimes (p \multimap O\alpha) \longrightarrow p \otimes (p \multimap O\alpha)}{\sim\sim p \otimes (p \multimap O\alpha) \longrightarrow O\alpha}\ (\mathrm{t}) \qquad \mathbf{3}\ \dfrac{\dfrac{}{p \multimap O\alpha \longrightarrow p \multimap O\alpha}\ (1)}{p \otimes (p \multimap O\alpha) \longrightarrow O\alpha}\ (\mathrm{cl})}\ (\mathrm{cut})$$

$$\mathbf{2}\ \dfrac{\dfrac{\vdots}{\sim\sim p \otimes (p \multimap O\alpha) \longrightarrow O\alpha} \qquad \dfrac{}{O\alpha \longrightarrow P_s\alpha}\ (\mathbf{D})}{\sim\sim p \otimes (p \multimap O\alpha) \longrightarrow P_s\alpha}\ (\mathrm{cut})$$

$$\mathbf{1}\ \dfrac{\dfrac{\dfrac{}{(p \multimap O\alpha)\otimes \sim\sim p \longrightarrow (p \multimap O\alpha)\otimes \sim\sim p}\ (1)}{(p \multimap O\alpha)\otimes \sim\sim p \longrightarrow \sim\sim p \otimes (p \multimap O\alpha)}\ (\mathrm{b}) \qquad \dfrac{\vdots}{\sim\sim p \otimes (p \multimap O\alpha) \longrightarrow P_s\alpha}}{(p \multimap O\alpha)\otimes \sim\sim p \longrightarrow P_s\alpha}\ (\mathrm{cut})$$

3.4 Decidability

In addition to the soundness and completeness theorems, string diagrams can also be used to show that $\mathcal{CNR}$ is decidable. The construction provided within this section

will be useful to determine the validity of conditional normative inferences. Consider the following definitions.

Definition 3.5 (Weak-consistency) *A list of formulas $\varphi_1, \ldots, \varphi_n$ is* weakly-consistent *if and only if there is no path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to 0 on $\mathcal{B} : \varphi_1, \ldots, \varphi_n$.*[11]

The contraposition of this definition yields the following definition of *weak-inconsistency*.

Definition 3.6 (Weak-inconsistency) *A list of formulas $\varphi_1, \ldots, \varphi_n$ is* weakly-inconsistent *if and only if there is a path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to 0 on $\mathcal{B} : \varphi_1, \ldots, \varphi_n$.*

Proposition 3.7 *Definitions 3.5 and 3.6 are logically equivalent.*

From the definition of weak-inconsistency, it trivially follows that:

Corollary 3.8 *If the list $\varphi, \sim \psi$ is weakly-inconsistent, then a path between $\varphi \otimes \sim \psi$ and 0 can be drawn on $\mathcal{B} : \varphi, \sim \psi$.*

Some remarks on weak-consistency are in order. Usually, the consistency of a *theory* or a *logical system* is defined via its theorems. For instance, one might say that a theory $\mathcal{T}$ is inconsistent when there is a formula φ such that both φ and $\sim \varphi$ are consequences of $\mathcal{T}$. Similarly, one could say that $\mathcal{T}$ is inconsistent when 0 (i.e., the constant representing falsehood) is a consequence of $\mathcal{T}$. This understanding, however, cannot be applied to our framework. Indeed, a logical system is not understood as a list of formulas (theorems) but is rather understood as a collection of (equivalence classes of) proofs. By opposition to the usual notion of consistency, weak-consistency is not the property of a deductive system. Instead of being the property of a collection of (equivalences classes of) proofs, weak-consistency is the property of a list of formulas.

The notion of weak-inconsistency can be distinguished from the usual understanding of inconsistency within Cartesian deductive systems (e.g., in intuitionistic or classical logic). In a Cartesian deductive system, the members of a conjunction can always be detached. From this property, one can see why weak-inconsistency is actually *weaker* than the usual understanding of inconsistency. In a Cartesian deductive system, a list $\varphi_1, \ldots, \varphi_n$ would be inconsistent as soon as there is a path from some members of the list to 0. This, however, does not apply to non-Cartesian deductive systems since the list must be taken altogether. For example, $\varphi \otimes 0$ is not weakly-inconsistent since 0 cannot be detached from the multiplicative conjunction. However, in a Cartesian deductive system, $\varphi \otimes 0$ is weakly-inconsistent.

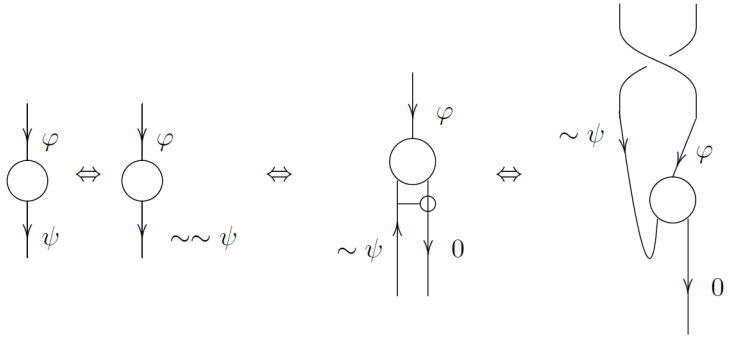
[11] Note that we are *not* using this terminology as it is used in consistency models or in [29]. Linear consistency might have been another significant appellation, but as *weak-consistency* it also already has a meaning in the literature. Monoidal consistency might also have been another candidate for the terminology, but we preferred to go with the more malleable (and meaningful) term *weak*.

Moreover, there is $0 \longrightarrow \varphi$ for any φ in a Cartesian deductive system. In monoidal logics, this property does not necessarily hold. Hence, it is noteworthy that weakly-inconsistent propositions are not necessarily isomorphic to 0 within monoidal deductive systems, while inconsistent propositions are isomorphic to 0 within Cartesian deductive systems. Therefore, it follows that weakly-inconsistent formulas within a monoidal deductive system are not necessarily logically equivalent, unlike inconsistent propositions within Cartesian deductive systems.[12]

The notion of weak-inconsistency is relevant to determine the validity of an inference given the following lemmas.

Lemma 3.9 *$f : \varphi \longrightarrow \psi$ is a $\mathcal{CNR}$-arrow if and only if $f^0 : \varphi \otimes \sim \psi \longrightarrow 0$ is.*
Proof. It follows from (cl) and the definition of negation. The proof from right to left requires the axiom for classical negation. ■

Lemma 3.10 *$f : \varphi \longrightarrow \psi$ is valid if and only if $\varphi, \sim \psi$ is weakly-inconsistent.*
Proof. Assume that f is valid. It follows that there is a path from $\varphi \otimes \sim \psi$ to 0 on every $\mathcal{B} : \varphi_1, \ldots, \varphi_n$.

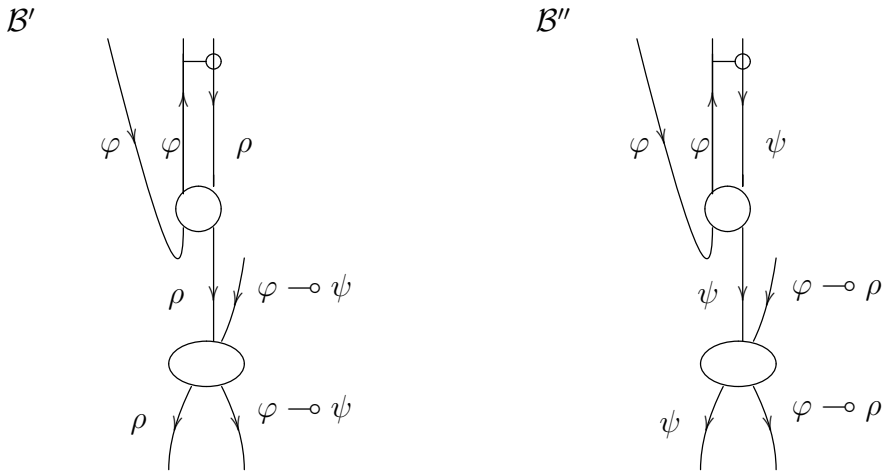


The converse is proven similarly. ■

We now provide a procedure to determine whether a list of formulas is weakly-inconsistent or not. First, let us introduce some terminology. Since different string diagrams can be drawn on a board $\mathcal{B} : \varphi_1, \ldots, \varphi_n$ depending on the order of the rules that are applied, we need to introduce the notion of a *page* on a board. The notation for the pages will be explained further in the next section, but for the moment let us write $\mathcal{B}'$, $\mathcal{B}''$, $\mathcal{B}'''$, etc., to refer to different pages of $\mathcal{B}$. Consider for example the board $\mathcal{B} : \varphi, \varphi \multimap \psi, \varphi \multimap \rho$. Many continuous strings can be drawn on that board using all the strings in the list. For instance, depending on whether we combine φ with $\varphi \multimap \psi$

[12]Similarly, tautologies are not logically equivalent within monoidal deductive systems, while they are within Cartesian deductive systems.

or $\varphi \multimap \rho$, we can have two continuous strings (without braiding) using all the strings in the list. In this case, we have the following pages.



We will say that a board $\mathcal{B} : \varphi_1, \ldots, \varphi_n$ *closes* when it has one page on which there is a path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to 0, that is, if there is a continuous string linking every member of the list to (and only to) 0.

Given the identities between the strings $\varphi \otimes \psi$ and $\psi \otimes \varphi$ and with φ and $\sim\sim \varphi$, we do not need to bother with the braiding rules and the double negation process. We will say that a list $\varphi_1, \ldots, \varphi_n$ is in its *standard form* when every double negation is erased and every tensor product of the form $\varphi \otimes \psi$ is written according to the same order. Moreover, since contraposition of conditional strings is satisfied (the proof is left to the reader), the conditionals must be written to facilitate composition. For example, the standard form of the list $\varphi, \psi \otimes \rho, \sim\sim \varphi \multimap (\rho \otimes \psi), \sim\sim\sim \tau, \sim \varphi \multimap\sim \sigma$ would be the list $\varphi, \psi \otimes \rho, \varphi \multimap (\psi \otimes \rho), \sim \tau, \sigma \multimap \varphi$.

Let $\mathcal{L}$ be a list and $\mathcal{L}^*$ be its standard form. If a string diagram can be drawn on a board $\mathcal{B} : \mathcal{L}$, then it can also be drawn on $\mathcal{B} : \mathcal{L}^*$, and reciprocally.

We now provide a procedure that will enable us to determine every continuous string that can be drawn from each and every member of the list.

Here are some basic rules of application.

- Each step is to be applied to only one string at a time.
- At each step, determine the number m of possible applications.
- If there are m strings to which the step can be applied, then do m copies of the page, each containing a possible application.
- Apply the procedure to each page.

The procedure is as follows.

Step 1 Rewrite the list in its standard form.

Step 2 Draw every string of the list in the hypotheses box.

Step 3 While the board remains open, copy the board on four different pages. Otherwise, if the board is closed, stop.

Step 3.1 On the first page, apply the rule for parallel process to obtain a continuous string joining all the strings on the board and then stop.

Step 3.1.1 If there is a path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to (and only to) 0, then close the board.

Step 3.2 On the second page, apply the string schema Ⓓ to one of the strings of the form $O\varphi$. Otherwise, if there is no such string, stop.

Step 3.2.1 If there is a path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to (and only to) 0, then close the board and stop. Otherwise, if the board remains open, repeat **Step 3**.

Step 3.3 On the third page, apply the string schema Ⓟ to one of the strings of the form $P_s\varphi$. Otherwise, if there is no such string, stop.

Step 3.3.1 If there is a path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to (and only to) 0, then close the board and stop. Otherwise, if the board remains open, repeat **Step 3**.

Step 3.4 On the fourth page, combine two formulas of the form φ and $\varphi \multimap \psi$ using Ⓒ. Otherwise, if there are no such strings, stop.

Step 3.4.1 If there is a path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to (and only to) 0, then close the board and stop. Otherwise, if the board remains open, repeat **Step 3**.

Step 3 is the recursive step that allows us to obtain each and every possible continuous string that can be formed using each and every member of the list $\varphi_1, \ldots, \varphi_n$. If the board is closed and the procedure is stopped, then $\varphi_1, \ldots, \varphi_n$ is weakly-inconsistent. Otherwise, if the board remains open and the procedure is stopped, then there is no path from $\varphi_1 \otimes \cdots \otimes \varphi_n$ to (and only to) 0, hence $\varphi_1, \ldots, \varphi_n$ is weakly-consistent.

From this procedure, we can list each and every continuous string that can be drawn on $\mathcal{B} : \varphi_1, \ldots, \varphi_n$ using every member of the list. Hence, it enables us to prove theorem 3.11.

Theorem 3.11 (Decidability) *For any arrow $f : \varphi \longrightarrow \psi$, we can show whether it is valid or not. Similarly, we can show whether f is derivable or not.*

Proof. Using the aforementioned procedure, we can determine whether $\varphi, \sim \psi$ is weakly-inconsistent or not. If it is, then f is valid. If it is not, then by lemma 3.10 it is invalid. ■

In the following section, we will provide an example of application of this procedure. For the moment, let us see why our proposal is relevant to the study of conditional normative reasoning.

4 Applying string diagrams to conditional reasoning

4.1 Motivations

As it was mentioned at the beginning of this paper, our goal is to use string diagrams to test the validity of conditional reasoning. In this respect, our initial intention was to adapt Baez's and Stay's work to critical thinking. Our proposal is inspired and motivated by Garson's [11] work, who used semantical trees to test the validity of modal inferences. Students in critical thinking courses are usually introduced to propositional, first-order and, sometimes, modal logics to model and analyze inferences. While it is known that propositional logic and the first-order calculus are insufficient to model normative reasoning (cf. [24]), it happens that the usual (deontic) modal systems such as K and KD face some serious problems when dealing with *conditional* normative reasoning. A conditional normative reasoning is understood as a normative inference within which there are conditional (and potentially conflicting) obligations.

The usual method in critical thinking to test the validity of an inference is to verify whether or not it possesses a counter-example (cf. [25]). To accomplish this, one must verify whether or not the premises are consistent with the negation of the conclusion. If they are, then the reasoning is invalid, and if they are not, then it is valid. Despite the important structural differences between $\mathcal{CNR}$ and the other approaches within the literature, our aim was to analyze conditional normative reasoning without developing a method that is too different from the usual tools in critical thinking, such as Garson's semantical trees. In this respect, we introduced the notion of weak-consistency to test the validity of normative inferences. With the help of Baez's and Stay's string diagrams, we obtain a method comparable to Garson's that allows to visually represent the structure of an argument.

It is well-known that a monadic standard system cannot properly model conditional normative reasoning without considering further modalities or assuming a primitive connective for conditional obligations. There are three major difficulties one faces when trying to model conditional normative reasoning. Consider a standard system such as KD for example (cf. [7]). The first obstacle is the problem of *factual detachment* of deontic conditionals (cf. [33]). Given an obligation $O\psi$ conditional to a context φ, this problem amounts to the fact that even though we generally want to allow for the detachment of the conditional obligation when the context is present, there can be other specific situations where we want to block the detachment of $O\psi$ given further relevant information. However, modeling conditional obligations through a monadic O within a

standard system implies that the following inference schema is satisfied. Hence, $O\psi$ is obtained as soon as φ is in the context (notwithstanding further relevant information that might thwart $O\psi$).

$$\frac{\varphi \qquad \varphi \supset O\psi}{O\psi}$$

The second difficulty is the problem of *augmentation* (cf. [15]), also known as the problem of *strengthening the antecedent* of a deontic conditional (cf. [1]). This problem happens when we try to model conditional obligations through a material conditional $\supset$. Indeed, it implies that the following inference pattern is always satisfied. Hence, if $O\psi$ is an obligation conditional to φ, then it is also conditional to any context that includes φ.

$$\frac{\varphi \supset O\psi}{(\varphi \wedge \rho) \supset O\psi}$$

The third problem concerns conflicting obligations and is known as *deontic explosion*. Given two conflicting obligations, it implies that anything is obligatory.

$$\frac{O\varphi \wedge O\neg\varphi}{O\psi}$$

These problems are usually presented as arguments in favor of non-monotonic or adaptive foundations for deontic logic (e.g., [14], [31] or [5]). It is usually assumed that these problems happen whenever one tries to model conditional normative reasoning without a primitive operator for deontic conditionals or further modalities. Recently, however, we showed in [27] that these problems occur when one models deontic conditionals via a monadic O within logics that satisfy the properties of a Cartesian closed deductive system. Such deductive systems can be proven to be instances of free Cartesian closed categories and are comparable from a proof-theoretical perspective to intuitionistic and classical logics. While augmentation and detachment are related to the fact that the usual conjunction $\wedge$ satisfies the properties of a categorical product, deontic explosion is a consequence of the fact that $\bot$ satisfies the properties of an initial object. We showed that conditional normative reasoning can be modeled via a monadic O insofar as the logic does not satisfy these properties.

The logic $\mathcal{CNR}$ was thus introduced in [27] as a foundational framework for conditional normative reasoning, able to model conditional normative inferences without facing augmentation, detachment or deontic explosion. This approach was then combined with an action logic (cf. [28]) and $\mathcal{CNR}$, as it is currently presented, was introduced

in [26]. We already discussed at length the relevance of this system to model conditional normative reasoning and we provided a detailed analysis of how it deals with the aforementioned problems and the paradoxes of deontic logic. Accordingly, we will not discuss these issues further and we refer the reader to these articles for a thorough presentation.

For the remaining of this section, we will provide an example of application of the decision procedure using Chisholm's [8] paradox, which will then be followed by an example of validity testing.

4.2 Chisholm's paradox

According to the results of the previous section, we have the following situation: representing an inference from a list of premises $\varphi_1, \ldots, \varphi_n$ to a conclusion ψ by a deduction arrow $\varphi_1 \otimes \cdots \otimes \varphi_n \longrightarrow \psi$, we can determine whether this inference is valid or not by determining if $\varphi_1, \ldots, \varphi_n, \sim \psi$ is weakly-inconsistent (or not). The notion of weak-consistency was thus introduced as a tool to facilitate the verification of an inference's validity using string diagrams.

Chisholm's [8] paradox is without a doubt the most famous illustration of conditional normative reasoning within the deontic logic literature. It has already been analyzed extensively within the framework of $\mathcal{CNR}$ in [26, 27], to which we refer the reader for a thorough discussion. The paradox results from the following list of sentences.

1. John ought to not lie.
2. If John lies, then he ought to tell that he lied.
3. John does not ought to tell that he lied if he does not lie.
4. John lies.

This is a paradox for many deontic logics given that these sentences seem consistent within the natural language while they are not within the formal ones.

To properly model Chisholm's paradox, one must first be able to provide an appropriate translation of these sentences. Although there is a negation within the first sentence, this negation is not a propositional one. Indeed, the first sentence does not mean that it is false that 'John ought to lie' (i.e., $\sim O\alpha$). Rather, it is an ought statement stating that John ought to 'not lie' (i.e., $O\alpha^*$). The second sentence was initially formulated by 'it ought to be that if John lies, then he tells that he lied' in Chisholm's presentation. This was mainly to avoid redundancy between the sentences and preserve their independence within the formal language. Since in $\mathcal{CNR}$ we do not face the redundancy problem (cf. [26, 27]), we can translate the second sentence by $\boldsymbol{\alpha} \multimap O\beta$. The third sentence can have two different translation within $\mathcal{CNR}$, depending on the

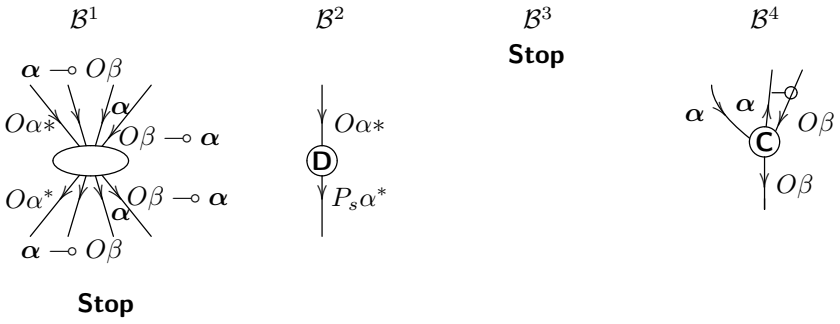
intended meaning of the negation. On the one hand, it can mean that 'if it is false that John lies, then it is also false that he ought to tell that he lied', which would be translated by $\sim \boldsymbol{\alpha} \multimap\!\!\sim O\beta$. On the other hand, it can also mean that 'if John does not lie, then it is false that he ought to tell that he lied', which would rather be translated by $\boldsymbol{\alpha^*} \multimap\!\!\sim O\beta$. The fourth sentence is simply translated by $\boldsymbol{\alpha}$.

To avoid the paradox, we must be able to model the fact that these sentences are weakly-consistent within $\mathcal{CNR}$. To do so, we must prove that there is no path from $O\alpha^* \otimes ((\boldsymbol{\alpha} \multimap O\beta) \otimes ((\sim \boldsymbol{\alpha} \multimap\!\!\sim O\beta) \otimes \boldsymbol{\alpha}))$ to 0 on $\mathcal{B} : O\alpha^*, \boldsymbol{\alpha} \multimap O\beta, \sim \boldsymbol{\alpha} \multimap\!\!\sim O\beta, \boldsymbol{\alpha}$

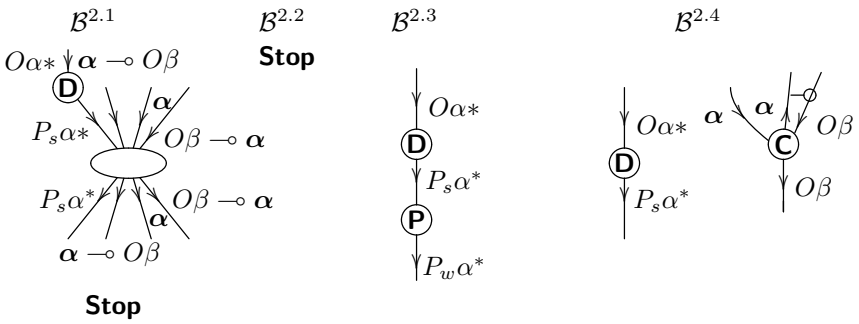
As a notational convention, we will name the pages according to the recursive procedure of **Step 3**. For example, $\mathcal{B}^1$ is the page containing the parallel process joining each and every string from the list, while $\mathcal{B}^{2.4}$ is the page resulting from applying Ⓓ first, and then Ⓒ.[13]

We now apply the procedure to determine whether the list $O\alpha^*, \boldsymbol{\alpha} \multimap O\beta, \sim \boldsymbol{\alpha} \multimap\!\!\sim O\beta, \boldsymbol{\alpha}$ is weakly-consistent or not. The first step consists in rewriting the list in its standard form: $O\alpha^*, \boldsymbol{\alpha} \multimap O\beta, O\beta \multimap \boldsymbol{\alpha}, \boldsymbol{\alpha}$.

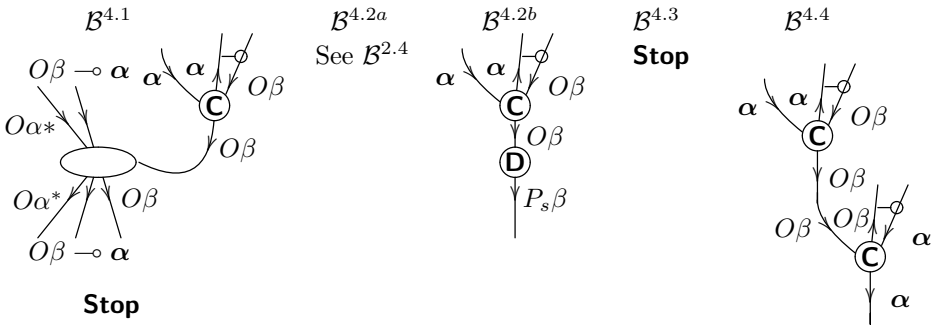
Loop 1



Loop 2

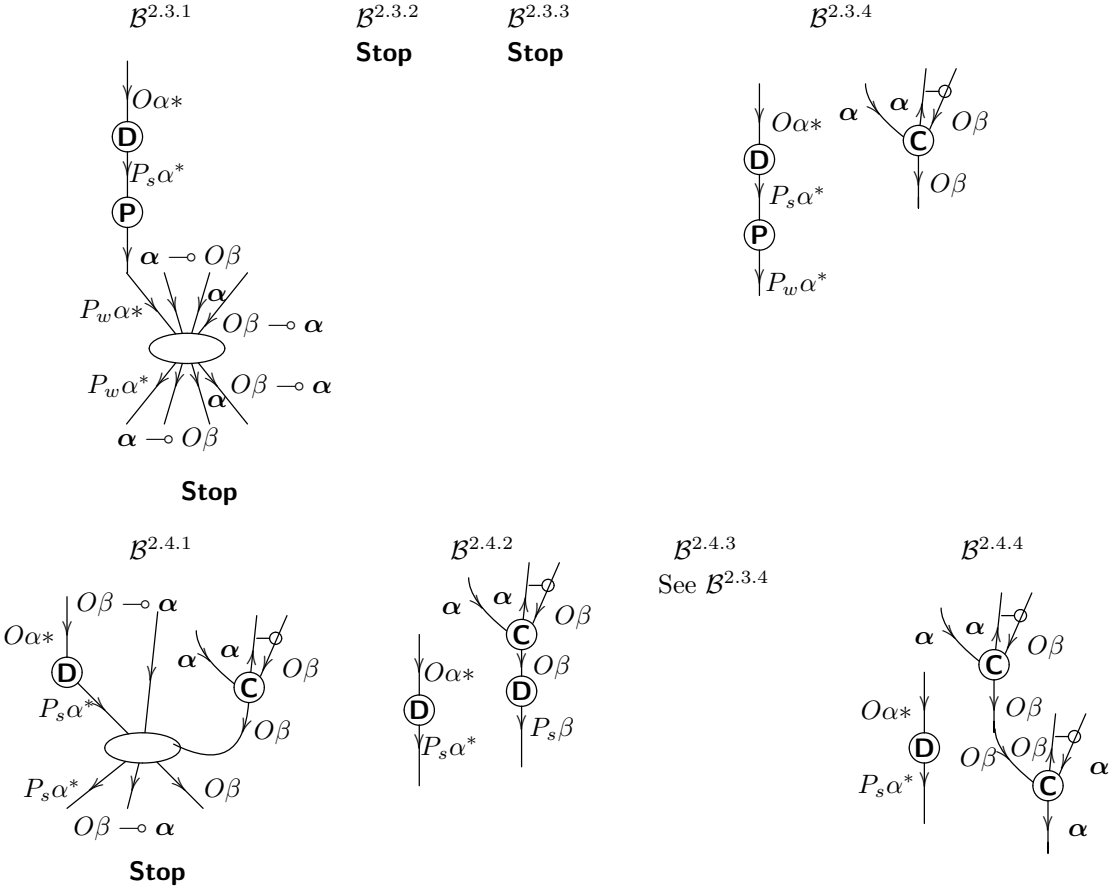


[13]If there are m copies of a page $\mathcal{B}^n$, then it can also be indexed by $i \leq m$ to indicate the appropriate page. In our case, we will write a, b, c, etc.

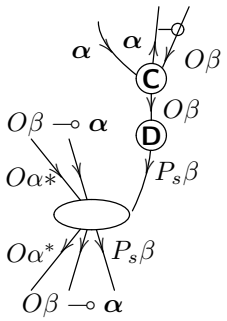


Note that on the page $\mathcal{B}^{4.2}$ there are two possible applications of Ⓓ. Hence, the page is copied on two different pages. The string diagram obtained on page $\mathcal{B}^{4.2a}$ is the same as the one on page $\mathcal{B}^{2.4}$. The same phenomenon appears below at **Loop 4**.

Loop 3



$\mathcal{B}^{4.2b.1}$

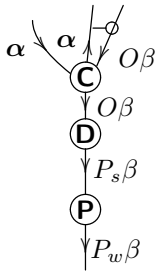


Stop

$\mathcal{B}^{4.2b.2}$

See $\mathcal{B}^{2.4.2}$

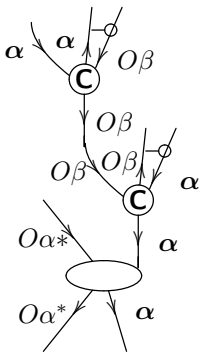
$\mathcal{B}^{4.2b.3}$



$\mathcal{B}^{4.2b.4}$

Stop

$\mathcal{B}^{4.4.1}$



Stop

$\mathcal{B}^{4.4.2}$

See $\mathcal{B}^{2.4.4}$

$\mathcal{B}^{4.4.3}$

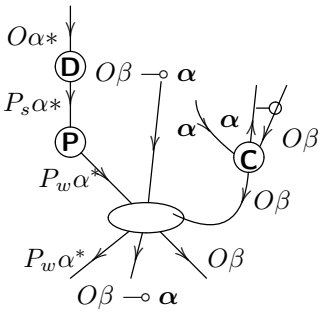
Stop

$\mathcal{B}^{4.4.4}$

Stop

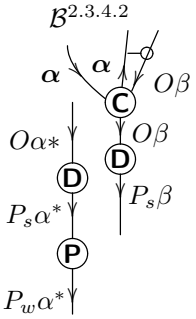
Loop 4

$\mathcal{B}^{2.3.4.1}$



Stop

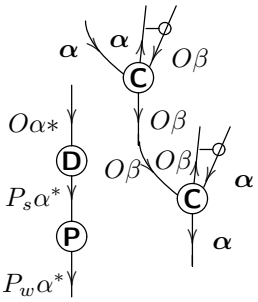
$\mathcal{B}^{2.3.4.2}$



$\mathcal{B}^{2.3.4.3}$

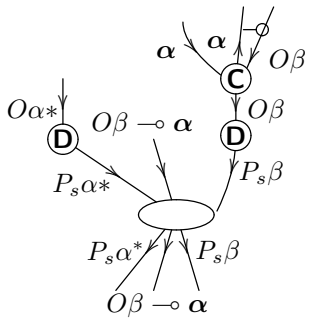
Stop

$\mathcal{B}^{2.3.4.4}$



$\mathcal{B}^{2.4.2.1}$ $\mathcal{B}^{2.4.2.2}$ $\mathcal{B}^{2.4.2.3a}$ $\mathcal{B}^{2.4.2.3b}$ $\mathcal{B}^{2.4.2.4}$

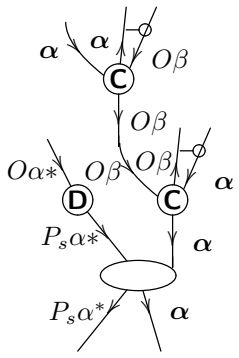
Stop See $\mathcal{B}^{2.3.4.2}$ See $\mathcal{B}^{4.2b.3.2}$ **Stop**



Stop

$\mathcal{B}^{2.4.4.1}$ $\mathcal{B}^{2.4.4.2}$ $\mathcal{B}^{2.4.4.3}$ $\mathcal{B}^{2.4.4.4}$

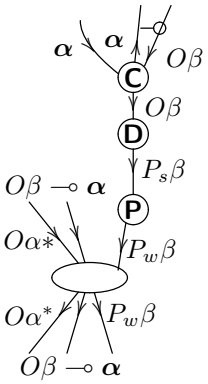
Stop See $\mathcal{B}^{2.3.4.4}$ **Stop**

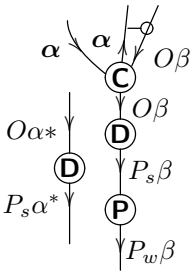


Stop

$\mathcal{B}^{4.2b.3.1}$ $\mathcal{B}^{4.2b.3.2}$ $\mathcal{B}^{4.2b.3.3}$ $\mathcal{B}^{4.2b.3.4}$

Stop **Stop**

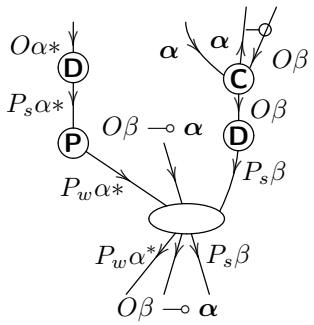




Stop

Loop 5

$\mathcal{B}^{2.3.4.2.1}$

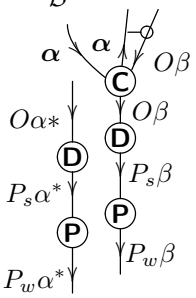


Stop

$\mathcal{B}^{2.3.4.2.2}$

Stop

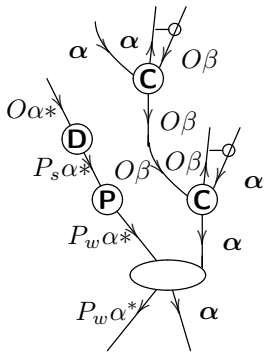
$\mathcal{B}^{2.3.4.2.3}$



$\mathcal{B}^{2.3.4.2.4}$

Stop

$\mathcal{B}^{2.3.4.4.1}$



Stop

$\mathcal{B}^{2.3.4.4.2}$

Stop

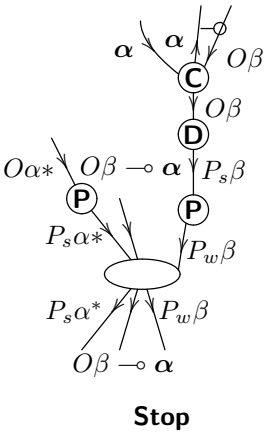
$\mathcal{B}^{2.3.4.4.3}$

Stop

$\mathcal{B}^{2.3.4.4.4}$

Stop

$\mathcal{B}^{4.2b.3.2.1}$



Stop

$\mathcal{B}^{4.2b.3.2.2}$

Stop

$\mathcal{B}^{4.2b.3.2.2}$

See $\mathcal{B}^{2.3.4.2.3}$

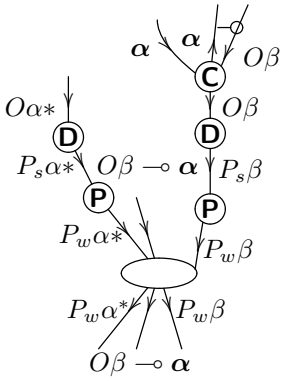
$\mathcal{B}^{4.2b.3.2.4}$

Stop

Loop 6

$\mathcal{B}^{2.3.4.2.3.1}$ $\mathcal{B}^{2.3.4.2.3.2}$ $\mathcal{B}^{2.3.4.2.3.3}$ $\mathcal{B}^{2.3.4.2.3.4}$

Stop **Stop** **Stop**



Stop

The procedure stops and we obtain each and every continuous string that can be formed using all (and only) the members of the list $O\alpha^*, \boldsymbol{\alpha} \multimap O\beta, O\beta \multimap \boldsymbol{\alpha}, \boldsymbol{\alpha}$. Note that if we had translated the third sentence using $\boldsymbol{\alpha^*} \multimap\!\sim O\beta$ instead of $\sim \boldsymbol{\alpha} \multimap\!\sim O\beta$, we would have obtained the same recursion loops with $\sim \boldsymbol{\alpha^*}$ instead of $\boldsymbol{\alpha}$. We thus have a proof that the list is weakly-consistent (since there is no path from the tensor product of the members of the list to 0), and hence Chisholm's paradox can be modeled within $\mathcal{CNR}$.

Incidentally, we can also see that $\mathcal{CNR}$ can model situations within which obligations have been violated. Carmo and Jones [6] argued that a logic which aims to model conditional obligations should be able to model contexts of violation.[14] On page $\mathcal{B}^{4.4.1}$, we have a path from the members of the list to $O\alpha^* \otimes \boldsymbol{\alpha}$: even though John lies, he shouldn't. In this case, we are able to model the fact that the obligation to not lie has been violated. Had we translated the third sentence using $\sim \boldsymbol{\alpha^*}$, we would obtain that John ought to not lie but it is false that he did not lie.

So far, we have been able to show that Chisholm's list of sentences is weakly-consistent within $\mathcal{CNR}$ and that contexts of violation can be modeled. To properly model Chisholm's paradox, another criterion is that $\mathcal{CNR}$ must be able to preserve the independence between the sentences (cf. [2]). This can be proven using the aforementioned procedure (the proof is left to the reader).[15]

That being said, one could also require that we should be able to derive from the list that John ought to tell that he lied. This can be derived in conjunction with other sentences (e.g., $\mathcal{B}^{2.4.1}$), but one might argue that it should be derivable alone: from Chisholm's list of sentences, it follows that there is a contrary-to-duty obligation for John to tell that he lied. Following van der Torre and Tan [32], Chisholm's paradox is a case of factual defeasibility, where the contrary-to-duty obligation for John to tell

[14] They also argued that it should be able to deal with the *pragmatic oddity*. See [27] for a discussion.
[15] Note that $1 \longrightarrow \boldsymbol{\alpha} \multimap (O\beta \multimap \boldsymbol{\alpha})$ is invalid within $\mathcal{CNR}$.

that he lied has priority over (but does not cancel) the obligation to not lie. Therefore, the obligation $O\beta$ holds under the circumstances described by the list. The contrary-to-duty obligation can thus be derived if we properly translate this phenomenon within $\mathcal{CNR}$'s language and specify that $O\beta$ holds under the context $O\alpha^* \otimes ((O\beta \multimap \boldsymbol{\alpha}) \otimes \boldsymbol{\alpha})$.[16]

To conclude this section, we will show how our string diagrammatic semantics can be used to test the validity of an inference. We will show that $O\beta$ can actually be derived if we properly translate the conditional obligation. Consider the following translation of Chisholm's sentences.

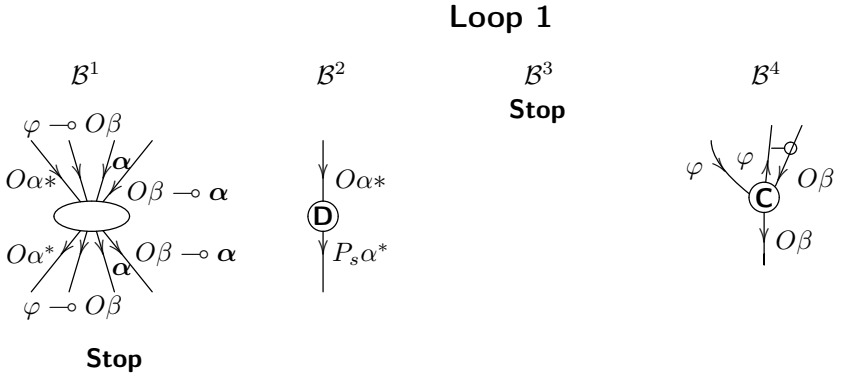
$$O\alpha^* \tag{1}$$
$$(O\alpha^* \otimes ((O\beta \multimap \boldsymbol{\alpha}) \otimes \boldsymbol{\alpha})) \multimap O\beta \tag{2}$$
$$\sim \boldsymbol{\alpha} \multimap \sim O\beta \tag{3}$$
$$\boldsymbol{\alpha} \tag{4}$$

We want to show that the following reasoning is valid.

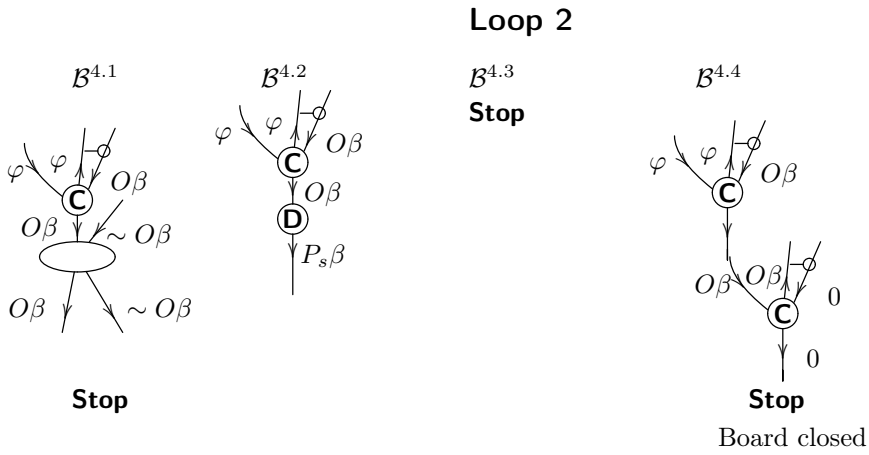
$$(O\alpha^* \otimes ((O\beta \multimap \boldsymbol{\alpha}) \otimes \boldsymbol{\alpha})) \otimes [(O\alpha^* \otimes ((O\beta \multimap \boldsymbol{\alpha}) \otimes \boldsymbol{\alpha})) \multimap O\beta] \longrightarrow O\beta$$

To accomplish this, we must determine whether the list $O\alpha^*, O\beta \multimap \boldsymbol{\alpha}, \boldsymbol{\alpha}$, $(O\alpha^* \otimes ((O\beta \multimap \boldsymbol{\alpha}) \otimes \boldsymbol{\alpha})) \multimap O\beta, \sim O\beta$ is weakly-consistent or not.

For visual clarity and due to space limitations, we will use φ instead of $O\alpha^* \otimes ((O\beta \multimap \boldsymbol{\alpha}) \otimes \boldsymbol{\alpha})$ in the following diagrams. At **Loop 2**, we will begin with the copies of page $\mathcal{B}^4$ given that it will close the board and stop the procedure. We thus obtain a path from (and only from) the members of the list to (and only to) 0, hence the proof of the previous inference's validity.



[16]For further discussion, see [27]. See also [26] for a discussion of Goble's [13] analysis of normative conflicts and conditional obligations.



5 Conclusion

Summing up, we applied Baez's and Stay's [3] string diagrams to the deontic logic $\mathcal{CNR}$. The deductive system $\mathcal{CNR}$ was initially developed to model conditional normative reasoning and was defined as a symmetric closed deductive system. This system was further developed in [26] and was defined on the grounds of an action logic $\mathcal{AL}$ (cf. [28]). Upgrading the string diagrammatic language of Baez and Stay [3], we provided a semantics for $\mathcal{CNR}$ and proved that it is sound and complete. Furthermore, we provided a decision procedure that allows to determine whether a proof is valid or not. This procedure was then applied to test the validity of conditional normative inferences and we showed how $\mathcal{CNR}$ can properly model Chisholm's paradox. Despite the differences between monoidal deductive systems and (the more familiar) intuitionistic or classical logic, the procedure can be compared to the usual methods that are employed to test the validity of inferences in critical thinking. Using the notion of weak-consistency, our string diagrammatic semantics enables us to determine whether an inference is valid or not by showing that the list including the premises and the negation of the conclusion is weakly-inconsistent (or not).

For future research, we intend to extend this method to the whole deontic deductive system presented in [26]. This will require that we adapt the diagrammatic language to deal simultaneously with various structures such as compact closed, $*$-autonomous and Cartesian categories. We also intend to study how this framework could be used in artificial intelligence. Recently, Pavlovic [23] used the language of string diagrams to define a *monoidal computer*. Basically, his idea is to conceive data as strings and programs as transformation processes. The string diagrammatic language he uses is the one of a symmetric monoidal category with the addition of operations for deleting, copying, and filtering, the latter two respecting the Frobenius condition. As such, there is common grounds between Pavlovic's monoidal computers and our approach. It will be interesting to see how our approach can be combined with Pavlovic's to program artificially intelligent agents, whose behaviors are regulated by norms.

Acknowledgment

I would like to thank Jean-Pierre Marquis for valuable comments and discussions. I am also in debt to the comments and suggestions of three anonymous referees for the current version of this article. This research was financially supported by the Social Sciences and Humanities Research Council of Canada.

References

[1] Carlos Alchourrón, *Detachment and defeasibility in deontic logic*, Studia Logica **57** (1996), no. 1, 5–18.

[2] Lennart Åqvist, *Deontic logic*, Handbook of Philosophical Logic (D. M. Gabbay and F. Guenthner, eds.), vol. 8, Kluwer Academic Publishers, 2nd ed., 2002, pp. 147–264.

[3] John C. Baez and Mike Stay, *Physics, topology, logic and computation: A Rosetta stone*, New Structures for Physics (B. Coecke, ed.), Lecture Notes in Physics, vol. 813, Springer, 2011, pp. 95–174.

[4] Micheal Barr, $*$*-autonomous categories*, Lecture Notes in Mathematics, vol. 752, Springer, 1979.

[5] Mathieu Beirlaen, Christian Straßer, and Joke Meheus, *An inconsistency-adaptive deontic logic for normative conflicts*, Journal of Philosophical Logic **42** (2013), no. 2, 285–315.

[6] José Carmo and Andrew J. I. Jones, *Deontic logic and contrary-to-duties*, Handbook of Philosophical Logic (D. M. Gabbay and F. Guenthner, eds.), vol. 8, Kluwer Academic Publishers, 2nd ed., 2002, pp. 265–343.

[7] Brian F. Chellas, *Modal logic: An introduction*, Cambridge University Press, 1980.

[8] Roderick Chisholm, *Contrary-to-duty imperatives and deontic logic*, Analysis **24** (1963), no. 2, 33–36.

[9] Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh, *Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus*, Annals of Pure and Applied Logic **164** (2013), no. 11, 1079–1100.

[10] N. Galatos, P. Jipsen, T. Kowalski, and H. Ono (eds.), *Residuated lattices: An algebraic glimpse at substructural logics*, Studies in Logic and the Foundations of Mathematics, vol. 151, Elsevier, 2007.

[11] James Garson, *Modal logic for philosophers*, Cambridge University Press, 2006.

[12] Jean-Yves Girard, *Linear Logic*, Theoretical Computer Science **50** (1987), no. 1, 1–102.

[13] Lou Goble, *Normative conflicts and the logic of 'ought'*, Noûs **43** (2009), no. 3, 450–489.

[14] John Horty, *Non-monotonic foundations for deontic logic*, Defeasible Deontic Logic (D. Nute, ed.), Kluwer Academic Publishers, 1997, pp. 17–44.

[15] Andrew J. I. Jones, *On the logic of deontic conditionals*, Ratio Juris **4** (1991), no. 3, 355–366.

[16] André Joyal and Ross Street, *The geometry of tensor calculus, II*, Available at `maths.mq.edu.au/ street/GTCII.pdf`.

[17] ———, *The geometry of tensor calculus, I*, Advances in Mathematics **88** (1991), 55–112.

[18] Gregory M. Kelly and Miguel L. Laplaza, *Coherence for compact closed categories*, Journal of Pure and Applied Algebra **19** (1980), 193–213.

[19] Joachim Lambek, *Deductive systems and categories I*, Mathematical Systems Theory **2** (1968), no. 4, 287–318.

[20] ———, *Deductive systems and categories II. Standard constructions and closed categories*, Category Theory, Homology Theory and their Applications I (P. J. Hilton, ed.), Lecture Notes in Mathematics, vol. 86, Springer, 1969, pp. 76–122.

[21] Joachim Lambek and Philip Scott, *Introduction to higher order categorical logic*, Cambridge University Press, 1986.

[22] Saunders Mac Lane, *Categories for the working mathematician*, 2nd ed., Springer, 1971.

[23] Dusko Pavlovic, *Monoidal computer I: Basic computability by string diagrams*, Information and Computation **226** (2013), 94–116.

[24] Clayton Peterson, *Normative inferences and validity: A heuristic*, Cogency **5** (2013), no. 1, 85–105.

[25] ———, *Pensée rationnelle et argumentation*, Presses de l'Université de Montréal, 2013.

[26] ———, *The categorical imperative: Category theory as a foundation for deontic logic*, Journal of Applied Logic **12** (2014), no. 4, 417–461.

[27] ———, *Contrary-to-duty reasoning: A categorical approach*, Logica Universalis **9** (2015), no. 1, 47–92.

[28] ———, *A logic for human actions*, Applications of Formal Philosophy (G. Payette and R. Urbaniak, eds.), Springer, 2015, To appear.

[29] Gemma Robles and José M. Méndez, *Strong paraconsistency and the basic constructive logic for an even weaker sense of consistency*, Journal of Logic, Language and Information **18** (2009), no. 3, 357–402.

[30] Peter Selinger, *A survey of graphical languages for monoidal categories*, New Structures for Physics (B. Coecke, ed.), Lecture Notes in Physics, vol. 813, Springer, 2011, pp. 289–355.

[31] Christian Straßer, *A deontic logic framework allowing for factual detachment*, Journal of Applied Logic **9** (2011), no. 1, 61–80.

[32] Leendert van der Torre and Yao-Hua Tan, *The many faces of defeasibility in defeasible deontic logic*, Defeasible Deontic Logic (D. Nute, ed.), Kluwer Academic Publishers, 1997, pp. 79–121.

[33] Job van Eck, *A system of temporally relative modal and deontic predicate logic and it's philosophical applications*, Logique et Analyse **25** (1982), no. 99, 249–290.

Clayton Peterson
Munich Center for Mathematical Philosophy
Ludwig-Maximilians Universität
Munich
E-mail: clayton.peterson@outlook.com