Categorical Continuation Semantics for Concurrency

Flavien Breuvart
Université Sorbonne Paris Nord, CNRS, Laboratoire d'Informatique de Paris Nord, LIPN, F-93430 Villetaneuse, France

Hugo Paquet
Inria, École Normale Supérieure – PSL, CNRS, Paris, France

Abstract

Continuation semantics for simple programming languages can be axiomatized as a dialogue category: a symmetric monoidal category equipped with a negation operation. This axiomatization makes clear the relationship between game semantics, CPS transformations, and continuation monads.

In this paper we extend dialogue categories with 2-categorical structure and concurrent primitives. This is inspired by a recent analysis of concurrency based on 2-categorical monads. We show that the fine-grained structure of dialogue categories, not generally available in other semantic models, can be exploited to give a type to concurrent primitives *join* and *fork*. Our main theorem is that this simple axiomatization induces a concurrent continuation 2-monad. We also show that this framework is expressive beyond call-by-value monadic programming.

The definitions in this paper are illustrated by concrete constructions in concurrent game semantics, and our results give a formal categorical basis for concurrent strategies. From a more practical perspective, our approach suggests a candidate target language for linear CPS transformations of concurrent programming languages.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Denotational semantics; Theory of computation → Concurrency; Theory of computation → Linear logic

Keywords and phrases denotational semantics, 2-categories, concurrency, continuations, game semantics

Digital Object Identifier 10.4230/LIPIcs.FSCD.2025.10

Acknowledgements We have benefited from discussing this work with many people including Morgan Rogers, Philip Saville, Tarmo Uustalu and Guannan Wei. Thanks to two anonymous referees for helpful feedback.

1 Introduction

Continuation models are special denotational models in which the semantics of types and terms reflects the shape of continuation-passing style (CPS) transformations.

Our purpose in this paper is to import a recent 2-categorical analysis of concurrency [33] to the theory of continuation models. Concretely, we look at continuation models in 2-categories, and axiomatize two combinators (fork and join) which can be used for parallel programming in combination with standard CPS constructs.

This work is inspired and motivated by existing constructions for concurrency in game semantics. Game semantics is a well-known source of continuation models, and game models often exhibit a linear structure which appears to be important for concurrency. We illustrate our concurrent combinators with a simple game semantics based on pomsets.

1.1 Dialogue categories: an abstract notion of linear continuation model

We focus on continuation models axiomatized as *dialogue categories* [26]. A dialogue category is a symmetric monoidal category equipped with an object $\bot$ such that a function space $A \multimap \bot$ exists for every object A. The object $\bot$ is a return type for continuations, and the condition reflects the fact that, after a CPS transformation, all functions should have return type $\bot$. In a dialogue category, the object $A \multimap \bot$ is typically written $\neg A$, and this is a functor $\neg : \mathcal{C} \to \mathcal{C}^{\mathrm{op}}$ which is involved in a "self-adjunction" $\neg \dashv \neg$. This induces a continuation monad $A \mapsto \neg\neg A$ on $\mathcal{C}$ and a continuation comonad $A \mapsto \neg\neg A$ on $\mathcal{C}^{\mathrm{op}}$. (The respective Kleisli and coKleisli categories are dual to each other. This is connected to the duality of call-by-value and call-by-name [38].)

It is significant that dialogue categories are taken to be monoidal rather than cartesian. This way, the theory includes models of linear continuations and linear CPS transformations (e.g. [14, 3]). Cartesian categories are a special case of monoidal categories, so standard continuation models are also included, but preliminary results suggest that to model concurrency like we do in this paper, a linear model is convenient (§4.3).

In this paper we extend dialogue categories in two steps: first we move to a generalized 2-categorical setting (we define *dialogue 2-categories*, §2.2), then we add concurrency (§3.2).

1.2 The 2-categorical nature of concurrent computation

A 2-categorical semantic model has objects representing types, morphisms representing programs or processes, and 2-cells (morphisms between morphisms). Invertible 2-cells often represent program symmetries (e.g. [11, 19, 7]), but it is not generally clear what non-invertible 2-cells represent.

Our purpose is to exploit the 2-cells for modelling concurrency. The motivation is that maps between concurrent processes are used for a diversity of models, including Petri nets, event structures, or asynchronous transition systems [39]. These maps typically encode a simulation of one process inside another.

A form of 2-cell also appears in the axiomatic approach to concurrency advocated by Hoare [15] (see also [16, 29]). In that framework, one has abstract operators for parallel ($\|$) and sequential ($;$) compositions of processes. one also has maps of (or, at least, a partial order on) processes, used to encode usual associativity, unitary and commutativity laws, as well as the weak interchange law between parallel and sequential composition:

$$(P \parallel Q); (R \parallel S) \longrightarrow (P; R) \parallel (Q; S) \tag{1}$$

As an example we look at the weak interchange law in pomsets, a very simple model of concurrent processes.

▶ **Definition 1.** *For a fixed set L, a* pomset labelled in L *is a finite partially ordered set $P = (|P|, \leq)$ equipped with a labelling function $|P| \to L$. A* configuration *of P is a down-closed subset. A* map of pomsets $P \to P'$ *is an injective function $|P| \to |P'|$ which preserves configurations and labelling.*

The informal idea is that L is a set of possible computational events, and elements of P are occurrences of these events. The partial order is a causal dependency relation: if $p \leq q$ in a pomset P, then q can only occur once p has occurred.

Pomsets support operations of parallel and sequential composition:

▶ **Proposition 2.** *For pomsets P and Q, define $P \parallel Q$ as the disjoint union (coproduct) of labelled posets, and define $P;Q$ as the disjoint union where additionally $(1,p) \leq_{P;Q} (2,q)$ for every $p \in |P|$ and $q \in |Q|$ (using the usual tagging for components of a disjoint union). There is a map of pomsets $(P \parallel Q);(R \parallel S) \longrightarrow (P;R) \parallel (Q;S)$, natural in P, Q, R, S.*

A concrete example is given by the map of pomsets $(\boxed{a} \rightarrow \boxed{b}) \longrightarrow (\boxed{a} \quad \boxed{b})$, for $a, b \in L$, where an arrow between pomset elements indicates a causal dependency in $\leq$. This map is an instance of Proposition 2 with $Q = R = \emptyset$, $P = (\boxed{a})$ and $S = (\boxed{b})$.

A *trace* of a pomset E is a word $t \in L^*$ viewed as a totally-ordered pomset, equipped with a map of pomsets $x \to t$ from a configuration $x \subseteq E$. It should be clear that for every map $E \to E'$ there is an inclusion of traces of E into traces of E'.

The starting point of this paper is a 2-categorical axiomatization, due to Rivas & Jaskelioff [35], and further developed in [33], which puts together the weak interchange law (1) and monadic models of program effects. In the present work we focus on the special case of continuation monads on dialogue categories, and observe that a concurrent 2-monad can be recovered using simple primitives for manipulating processes (join and fork, §3.2).

1.3 Contributions of this paper

This paper combines ideas from two lines of research: linear continuation models in dialogue categories, and 2-categorical models for concurrency. It is organized as follows:

- In §2, we recall basic notions of 2-category theory, and we show that properties of dialogue categories generalize to this setting. For a toy language, we define call-by-value and call-by-name semantics in a dialogue 2-category. We illustrate this with a simple 2-category from game semantics.
- In §3, we introduce our notion of concurrent dialogue 2-category (Definition 20) and show that it gives a concurrent continuation 2-monad for a parallel call-by-value semantics (Theorem 22). We also suggest a possible use case for parallelism in call-by-name (§3.4).
- In §4 we discuss concrete models, degenerate models, and we give a brief syntactic perspective with a concurrent CPS target language (§4.4).

We motivate our definitions using examples and the toy calculus, but these are intended to illustrate the basic ideas rather than the full power of the approach. In particular, our language is linear and has no side effects, so issues of call-by-name and call-by-value may seem irrelevant, but our methods will scale to more expressive languages and advanced continuation models.

Our high-level objective is to provide a new bridge between the categorical semantics of effects and the vast literature on models for concurrency. This work also brings new insights to game semantics: we show that the causal patterns (forks and joins) found in complex concurrent models can be given an algebraic basis.

2 Dialogue categories with 2-dimensional structure

Our first step is to generalize the notion of dialogue category to a 2-categorical setting. In this section we recall the basics of 2-category theory (§2.1), then we define a notion of dialogue 2-category (§2.2), and finally we show that 2-dimensional semantic structures (in particular, a strong continuation 2-monad) can be derived just like in the ordinary 1-categorical setting (§2.3).

2.1 Basic notions of 2-category theory

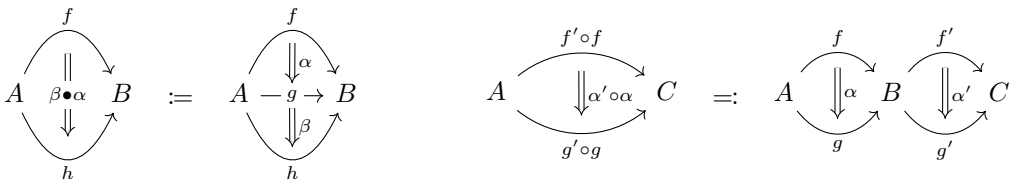
We give a short introduction to the main concepts. For a textbook reference, see [18].

▶ **Definition 3.** *A* 2-category $\mathcal{C}$ *consists of:*

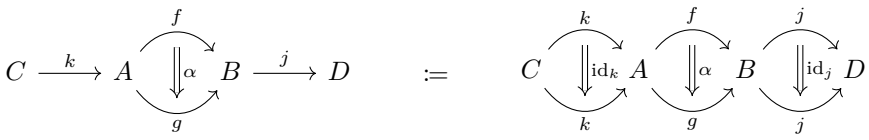
- *a collection* $|\mathcal{C}|$ *of* objects*;*
- *for every* $A, B \in |\mathcal{C}|$*, a category* $\mathcal{C}(A, B)$ *whose objects are called* 1-cells *and whose morphisms are called* 2-cells*;*
- *composition functors* $\circ_{A,B,C} : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$ *simply denoted* $\circ$*; and*
- *identity 1-cells* $\mathrm{id}_A \in \mathcal{C}(A, A)$*.*

Compositions and identities are subject to associativity and unit axioms, as expected.

Diagrams. A 2-category has an underlying 1-category, obtained by forgetting the 2-cells. Thus, in particular, we can reason about equality of 1-cells using ordinary commutative diagrams. We can also use diagrams to depict "vertical" and "horizontal" compositions of 2-cells: vertical composition (on the left below) refers to the composition of morphisms $\alpha : f \to g$ and $\beta : g \to h$ in the category $\mathcal{C}(A, B)$, while horizontal composition (on the right) refers to the image under $\circ_{A,B,C}$ of the morphism $(\alpha, \alpha') : (f, f') \to (g, g')$ in the product category $\mathcal{C}(A, B) \times \mathcal{C}(B, C)$:



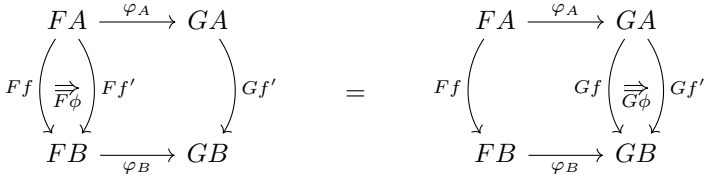
Horizontal composition of 2-cells induces the "whiskering" construction, in which a 2-cell is precomposed or postcomposed with a 1-cell. This is defined by horizontal composition with identity 2-cells:



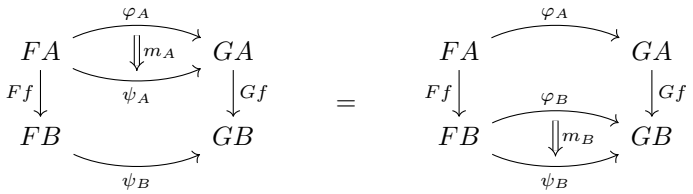
Functors, transformations, and modifications. The main concepts of 2-category theory are given in the definition below:

▶ **Definition 4.**

- *A* 2-functor $F : \mathcal{C} \to \mathcal{D}$*, between 2-categories* $\mathcal{C}$ *and* $\mathcal{D}$*, consists of a map of objects* $|\mathcal{C}| \to |\mathcal{D}|$*, and a collection of functors* $\mathcal{C}(A, B) \to \mathcal{D}(FA, FB)$*, preserving identity and composition.*
- *A* 2-natural transformation $\varphi : F \to G$ *between 2-functors* $F, G : \mathcal{C} \to \mathcal{D}$ *is a family of 1-cells* $\varphi_A : FA \to GA$*, indexed by objects* $A \in |\mathcal{C}|$*, satisfying the usual naturality axiom for 1-cells, and the following axiom for all 2-cells* $\phi : f \to f'$*:*



- *A* modification $m : \varphi \to \psi$ *between 2-natural transformations* $\varphi, \psi : F \to G : \mathcal{C} \to \mathcal{D}$ *is a family of 2-cells* $m_A : \varphi_A \to \psi_A$*, indexed by objects* $A \in |\mathcal{C}|$*, subject to the equation below:*



Finally, there are 2-categorical versions of symmetric monoidal categories and monads, known as *symmetric monoidal 2-categories* and *2-monads*. The definitions are the same except that the functors involved must be 2-functors and the natural transformations must be 2-natural transformations. The coherence axioms are all the same.

Remark on bicategories. The 2-categories in this paper are assumed to be *strict*, in the sense that the associativity and identity axioms of composition require actual equality of 1-cells. A more general notion is that of bicategory (sometimes weak 2-category) where the laws hold only up to invertible 2-cells, but then additional coherence axioms are needed on those 2-cells, and this significantly complicates the theory.

Many models encountered in practice are bicategories (e.g. [10, 4, 1]) and so this level of generality is very important. But every bicategory is biequivalent to a strict 2-category [21], so in this paper we keep everything strict to focus on the key conceptual points and not the coherence issues.

2.2 Dialogue 2-categories

We sketched the definition of dialogue category in §1. Here we give a formal definition of dialogue 2-category, which generalizes the 1-categorical definition [26, 25].

▶ **Definition 5.** *Let* $(\mathcal{C}, \otimes, I)$ *be a symmetric monoidal 2-category and* A, B *be objects of* $\mathcal{C}$*. An* internal hom from A to B *is an object* $A \multimap B \in \mathcal{C}$ *together with an isomorphism of categories* $\mathcal{C}(C \otimes A, B) \cong \mathcal{C}(C, A \multimap B)$ *which is natural in* C*.*

▶ **Definition 6.** *In a symmetric monoidal 2-category* $(\mathcal{C}, \otimes, I)$*, a* return object *is an object* $\bot \in \mathcal{C}$ *equipped with a choice of internal hom* $A \multimap \bot$*, for every object* $A \in \mathcal{C}$*.*

A symmetric monoidal 2-category is *closed* when it has all internal homs $A \multimap B$, and in this case any object can be a return object. By contrast, in a dialogue 2-category we require only one return object $\bot$. This is sufficient to model terms in continuation-passing style, using $\bot$ as the return type of continuations.

▶ **Definition 7.** *A* dialogue 2-category *is a pair consisting of a symmetric monoidal 2-category* $(\mathcal{C}, \otimes, I)$ *and a return object* $\bot \in \mathcal{C}$*.*

In the logical analysis of continuations, it makes sense to regard the construction $- \multimap \bot$ as a form of negation [13], and as is common we write $\neg A$ for $A \multimap \bot$.

An important observation is that negation induces a contravariant 2-functor on $\mathcal{C}$. This is expressed in terms of the opposite 2-category $\mathcal{C}^{\mathrm{op}}$, defined to have reversed 1-cells but not 2-cells: $\mathcal{C}^{\mathrm{op}}(A, B) \coloneqq \mathcal{C}(B, A)$.

▶ **Proposition 8.** *In a dialogue 2-category* $(\mathcal{C}, \otimes, I, \bot)$*, the operation* $A \longmapsto \neg A$ *extends to a 2-functor* $\neg : \mathcal{C}^{\mathrm{op}} \longrightarrow \mathcal{C}$*.*

Proof. For objects A and B, the action of the 2-functor $\neg$ on $\mathcal{C}(A,B)$ is defined by a composition of functors

$$\mathcal{C}(A,B) \xrightarrow{\mathrm{ev}_{B,I} \circ (\mathrm{id}_{B \multimap \bot} \otimes -)} \mathcal{C}((B \multimap \bot) \otimes A, \bot) \xrightarrow{\cong} \mathcal{C}(B \multimap \bot, A \multimap \bot),$$

where $\mathrm{ev}_{B,I} : (B \multimap \bot) \otimes B \longrightarrow \bot$ is the image of $\mathrm{id}_{B \multimap \bot}$ under the isomorphism $\mathcal{C}(B \multimap \bot, B \multimap \bot) \cong \mathcal{C}((B \multimap \bot) \otimes B, \bot)$. ◀

A basic property of a dialogue 2-category is the existence of a 2-adjunction between the negation 2-functor and its opposite:

$$\mathcal{C} \underset{\neg^{\mathrm{op}}}{\overset{\neg}{\underset{\bot}{\rightleftarrows}}} \mathcal{C}^{\mathrm{op}} \tag{2}$$

This is expressed more concretely by a family of natural isomorphisms $\mathcal{C}(A, \neg B) \cong \mathcal{C}^{\mathrm{op}}(\neg A, B) = \mathcal{C}(B, \neg A)$ between categories of 1-cells in $\mathcal{C}$.

These isomorphisms are in fact induced by a stronger parametrized family of isomorphisms, which can be regarded as the essential property of a dialogue 2-category:

▶ **Lemma 9.** *In a dialogue 2-category $(\mathcal{C}, \otimes, I, \bot)$, there is a family of isomorphisms of categories $\Phi^C_{A,B} : \mathcal{C}(C \otimes A, \neg B) \xrightarrow{\cong} \mathcal{C}(C \otimes B, \neg A)$ natural in A, B, C.*

The adjunction (2) is deduced by setting $C = I$, and applying the monoidal unit isomorphisms. We abuse notation and write $\Phi_{A,B}$ for the resulting isomorphism $\mathcal{C}(A, \neg B) \xrightarrow{\cong} \mathcal{C}(B, \neg A)$.

2.3 Semantics in a dialogue 2-category

In the 1-categorical setting, the situation of Lemma 9 gives what is sometimes known as a *strong* adjunction, because the monad induced by the adjunction is automatically strong [22].

A strong adjunction is a fine-grained categorical structure which induces a semantic model for call-by-value and one for call-by-name. To illustrate this in the context of dialogue 2-categories we consider a very simple lambda-calculus with linear types:

$$\begin{array}{l} A, B ::= I \mid A \multimap B \\ s, t ::= () \mid x \mid \lambda x.t \mid t\ s \end{array} \qquad \frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B} \qquad \frac{\Gamma \vdash t : A \multimap B \qquad \Delta \vdash s : A}{\Gamma, \Delta \vdash t\ s : B}$$

together with an axiom $\vdash () : I$. We now give the call-by-value and call-by-name continuation semantics. (This is closely related to existing translations of the calculus into linear call-by-push-value [8] or tensorial logic [30]. See the end of this section for further details.)

Call-by-value semantics. In the categorical semantics of call-by-value, the main ingredients are a strong monad and Kleisli exponentials [31].

▶ **Lemma 10** (Continuation 2-monad)**.** *For a dialogue 2-category $(\mathcal{C}, \otimes, I, \bot)$, the endofunctor on $\mathcal{C}$ given by the composition $\mathcal{C} \xrightarrow{\neg} \mathcal{C}^{\mathrm{op}} \xrightarrow{\neg} \mathcal{C}$ has the structure of a strong 2-monad.*

The only novelty is the 2-dimensional structure, but everything is completely analogous to the standard setting. The strong 2-monad structure is as follows:

$$\begin{array}{ll} \eta_A : A \to \neg\neg A & \eta_A = \Phi_{\neg A, A}(\mathrm{id}_{\neg A}) \\ \mu_A : \neg\neg\neg\neg A \to \neg\neg A & \mu_A = \neg\eta_{\neg A} \\ t_{A,B} : A \otimes \neg\neg B \to \neg\neg(A \otimes B) & t_{A,B} = \Phi^A_{\neg(A \otimes B), \neg\neg B}(\eta_{\neg B} \circ \Phi^A_{B, \neg(A \otimes B)}(\eta_{A \otimes B})) \end{array}$$

▶ **Lemma 11** (Kleisli exponentials). *For objects A and B of a dialogue 2-category $(\mathcal{C}, \otimes, I, \bot)$, the object $\neg(A \otimes \neg B)$ is an internal hom from A to $\neg\neg B$ in $\mathcal{C}$. In particular we have an evaluation 1-cell* $\mathrm{ev}_{A,\neg B} : \neg(A \otimes \neg B) \otimes A \to \neg\neg B$.

Proof. For every X, $\mathcal{C}(X, \neg(A \otimes \neg B)) \cong \mathcal{C}(X \otimes A \otimes \neg B, \bot) \cong \mathcal{C}(X \otimes A, \neg\neg B)$. ◀

Given a strong 2-monad and Kleisli exponentials, the semantics of the calculus is standard: a type defines an object of $\mathcal{C}$, inductively as $[\![I]\!]^v = I$ and $[\![A \multimap B]\!]^v = \neg([\![A]\!]^v \otimes \neg[\![B]\!]^v)$. Then a term $x_1 : A_1, \dots, x_n : A_n \vdash t : B$ defines a morphism $[\![t]\!]^v : [\![A_1]\!]^v \otimes \cdots \otimes [\![A_n]\!]^v \to \neg\neg[\![B]\!]^v$, as in the following (we use the same contexts and types as in the rules above): a variable x is interpreted as the unit $\eta_{[\![A]\!]^v}$, an abstraction $\lambda x.t$ is interpreted as the image of $[\![t]\!]^v$ under the isomorphism

$$\mathcal{C}([\![A_1]\!]^v \otimes \cdots \otimes [\![A_n]\!]^v \otimes [\![A]\!]^v, \neg\neg[\![B]\!]^v) \cong \mathcal{C}([\![A_1]\!]^v \otimes \cdots \otimes [\![A_n]\!]^v, \neg([\![A]\!]^v \otimes \neg[\![B]\!]^v)),$$

post-composed with the unit $\eta_{\neg([\![A]\!]^v \otimes \neg[\![B]\!]^v))}$, and finally an application $t\ s$ is interpreted as

$$\begin{aligned}[\![\Gamma]\!]^v \otimes [\![\Delta]\!]^v &\xrightarrow{[\![t]\!]^v \otimes [\![s]\!]^v} \neg\neg\neg([\![A]\!]^v \otimes \neg[\![B]\!]^v) \otimes \neg\neg[\![A]\!]^v \\ &\xrightarrow{\blacktriangleleft\blacktriangleright} \neg\neg(\neg([\![A]\!]^v \otimes \neg[\![B]\!]^v) \otimes [\![A]\!]^v) \xrightarrow{\neg\neg\mathrm{ev}} \neg\neg\neg\neg[\![B]\!]^v \xrightarrow{\mu_B} \neg\neg[\![B]\!]^v\end{aligned}$$

where for the arrow labelled ($\blacktriangleleft\blacktriangleright$) we must choose between left-right or right-left evaluation order.

Call-by-name semantics. The semantics of call-by-name takes place in the coKleisli 2-category for the 2-comonad $\neg\neg$ on $\mathcal{C}^{\mathrm{op}}$ [22]. We use an equivalent presentation of this coKleisli 2-category: we interpret types as objects of $\mathcal{C}$, with $[\![I]\!]^n = \neg I$ and $[\![A \multimap B]\!]^n = \neg[\![A]\!]^n \otimes [\![B]\!]^n$. There is an evaluation map $\mathrm{ev}_{\neg[\![A]\!]^n, [\![B]\!]^n} : \neg(\neg[\![A]\!]^n \otimes [\![B]\!]^n) \otimes \neg[\![A]\!]^n \to \neg[\![B]\!]^n$ induced by the dialogue structure. Each term judgement $x_1 : A_1, \dots, x_n : A_n \vdash t : B$ is interpreted as a 1-cell $\neg[\![A_1]\!]^n \otimes \cdots \otimes \neg[\![A_n]\!]^n \to \neg[\![B]\!]^n$ in $\mathcal{C}$. For the judgements in the type system above, a variable $x : A$ gives an identity morphism on $\neg[\![A]\!]^n$, an abstraction $\lambda x.t$ is interpreted using the isomorphism

$$\mathcal{C}(\neg[\![A_1]\!]^n \otimes \cdots \otimes \neg[\![A_n]\!]^n \otimes \neg[\![A]\!]^n, \neg[\![B]\!]^n) \cong \mathcal{C}(\neg[\![A_1]\!]^n \otimes \cdots \otimes \neg[\![A_n]\!]^n, \neg(\neg[\![A]\!]^n \otimes [\![B]\!]^n)),$$

applied to $[\![t]\!]^n$, and finally an application $t\ s$ is interpreted as

$$[\![\Gamma]\!]^n \otimes [\![\Delta]\!]^n \xrightarrow{[\![t]\!]^n \otimes [\![s]\!]^n} \neg(\neg[\![A]\!]^n \otimes [\![B]\!]^n) \otimes \neg[\![A]\!]^n \xrightarrow{\mathrm{ev}_{\neg[\![A]\!]^n, [\![B]\!]^n}} \neg[\![B]\!]^n. \tag{3}$$

As an aside, we note a duality which is the hallmark of continuation semantics (e.g. [38]):

▶ **Proposition 12** (Duality of call-by-value and call-by-name). *For a dialogue category $(\mathcal{C}, \otimes, I, \bot)$, there is an isomorphism of 2-categories*

$$\mathcal{K}\ell(\neg^{\mathrm{op}} \circ \neg) = \mathrm{co}\mathcal{K}\ell(\neg \circ \neg^{\mathrm{op}})^{\mathrm{op}}$$

for the Kleisli and coKleisli constructions associated with the induced 2-monad on $\mathcal{C}$ and 2-comonad on $\mathcal{C}^{\mathrm{op}}$, respectively.

Connections to tensorial logic and call-by-push-value. Dialogue categories are semantic models for *tensorial logic*, a linear sequent calculus involving only connectives $\otimes$ and $\neg$. Our call-by-value and call-by-name semantics correspond precisely to the standard translation of intuitionistic linear logic (restricted to our simple language) into tensorial logic [30].

In a different direction, the strong adjunction induced by a dialogue category is a model for a linear version of call-by-push-value [22, 8], and the call-by-name and call-by-value semantics correspond to translations into linear call-by-push-value. An interesting research direction is towards general 2-categorical strong adjunctions and models for CBPV, but this would require enriched or "locally graded" 2-categories. We can avoid this in the paper because continuation models are a very special case.

2.4 Illustration in a concrete model

To illustrate the role of 2-cells in continuation models we look at an example from game semantics. This is a deterministic game model based on pomsets, in which the 2-cells are maps between strategies. (We have chosen as simple an example as possible: our model does not support basic features like sum types, non-linear types, recursion, or nondeterminism. But we emphasize that the abstract structures in this paper also cover the more sophisticated game models in the literature.)

A simple game semantics. We consider a concurrent game semantics based on pomsets. This example serves to illustrate the 2-categorical extension of dialogue categories as well as the concurrent extension discussed in §4.1. This is a simplified deterministic and finitary version of the model in [34], which we give in terms of pomsets (i.e. finite partial orders, recall Definition 1) labelled in the two-element set $\mathbf{P} \coloneqq \{-, +\}$. For a pomset A, the pomset $A^\perp$ has the same underlying poset but the polarity labelling is flipped.

▶ **Definition 13.** *A* game *is a pomset $A = (|A|, \leq_A)$ (all pomsets are labelled in $\mathbf{P}$), which is* negative *in the sense that all minimal elements are labelled by $-$.*

A strategy *$A \to B$ is a negative pomset $S = (|S|, \leq_S)$, equipped with a map of pomsets $p_S : S \to A^\perp \parallel B$ satisfying a lifting condition.*[1] *A* map of strategies *from (S, p_S) to $(S', p_{S'})$ is a map of pomsets $S \to S'$ which commutes with the maps to $A^\perp \parallel B$.*

Strategies $A \to B$ and $B \to C$ compose by playing different roles in the middle game B, and there is an identity $A \to A$ which forwards moves from one copy to the next [34]. This gives a bicategory, but to bypass the coherence issues we define a 2-category $\mathcal{G}$ of games, such that each $\mathcal{G}(A, B)$ is a skeleton of the category of strategies $A \to B$ and maps between them.

(*Note:* using skeletons of hom categories does not generally work as a method for building a 2-category out of a bicategory. Thanks to the anonymous reviewer for emphasizing this. It does work for our example, because the bicategory has no non-identity 2-automorphisms. For more general game models, coherence issues need to be properly addressed.)

▶ **Proposition 14.** *The 2-category $\mathcal{G}$ has a symmetric monoidal structure induced by the operation $\parallel$ on pomsets. The monoidal unit is the empty game, denoted I. The singleton game $\perp \coloneqq \{*\}$ has the structure of a return object in $(\mathcal{G}, \parallel, I)$.*

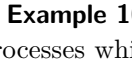
[1] We state the condition for the record, although it plays no role in the paper: for a configuration x of S, if either $p_S x \subseteq^- y$ or $y \supseteq^+ p_S x$ for a configuration y of $A^\perp \parallel B$ (where the polarity is a condition on all events in the set difference), then there should be a unique lifting of y to S.

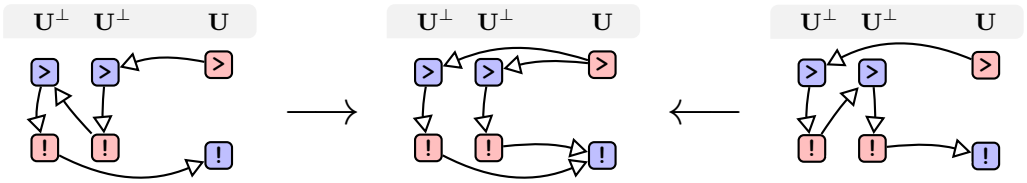
In $\mathcal{G}$, the construction $\neg A$ takes $A^\perp$ and appends a minimal negative move $*$ (cf. models in [23, 24]). We omit the full proof but illustrate the dialogue structure for singleton games.

▶ **Example 15** (Negation in games)**.** For games $A = \{a\}$ and $B = \{b\}$, the pomsets below are $(A \parallel B)^\perp \parallel \perp$ and $A^\perp \parallel \neg B$. In diagrams, positive moves are coloured in blue and negative moves in red, and the arrow represents the only nontrivial part of the partial order $(* \leq b)$.

Strategies on the left-hand game are also strategies on the right-hand game: by negativity of the pomsets, in a strategy S of type $A \parallel B \to \perp$, the element whose image is $*$ must necessarily occur first according to the order in S. Therefore, the extra constraint imposed by the arrow in the right-hand game is always obeyed in S, and thus S can also be given the type $A \to \neg B$. The converse property is immediate, so we have a correspondence, which extends easily to maps of strategies, i.e. there is an isomorphism $\mathcal{G}(A \parallel B, \perp) \cong \mathcal{G}(A, \neg B)$.

Applications of 2-cells.

▶ **Example 16** (Sequential and parallel execution)**.** Consider the game $\mathbf{U} =$  , modelling processes which are started by an external input and then eventually send a termination signal. (There are only two possible strategies: one for the process that sends a signal, and one for the process that doesn't.) Now, given two input processes of this type, we can run one after the other, or both in parallel. Corresponding to this there are three strategies $\mathbf{U} \parallel \mathbf{U} \to \mathbf{U}$, and 2-cells between them, relating the possible behaviours:



▶ Remark 17. Our focus is on concurrency, but some other applications of 2-cells in semantics are already well-established: models for recursion are typically cpo-enriched categories, which are a special case of 2-categories. In various examples the order is even defined by the existence of a structure-preserving embedding [32], making the 2-categorical aspects explicit. Similarly in models of non-determinism one typically has an embedding or inclusion $[\![t]\!] \hookrightarrow [\![t \oplus t']\!]$, where $\oplus$ is binary nondeterministic choice. The 2-categorical setup, in which we can compose 2-cells horizontally and vertically, enables compositional reasoning about possible returned values.

In summary, we have introduced a notion of dialogue 2-category, in which one can give continuation semantics in call-by-value or call-by-name. This is just like in the 1-dimensional case, but now 2-cells are included and can be manipulated in combination with the logical structure of the model. This is the setting we propose for concurrent continuation models.

3 Concurrent combinators for dialogue 2-categories

In this section we define an extended notion of dialogue 2-category with additional concurrent structure based on two families of combinators fork and join, typed as follows:

$$\mathrm{fork}_{A,B} : \neg A \otimes \neg B \longrightarrow \neg(A \otimes B) \qquad \mathrm{join}_{A,B} : \neg(A \otimes B) \longrightarrow \neg A \otimes \neg B$$

The definition will be set up to ensure, in particular, that the composite 1-cells

$$\neg\neg A \otimes \neg\neg B \xrightarrow{\text{fork}_{\neg A,\neg B}} \neg(\neg A \otimes \neg B) \xrightarrow{\neg\,\text{join}_{A,B}} \neg\neg(A \otimes B)$$

give $\neg\neg$ the structure of a *concurrent 2-monad* [35, 33, Definition 18 below]. Although this looks similar to the data for a monoidal monad, the 2-dimensional properties are weaker. (A continuation monad is only monoidal in very degenerate cases, as noticed by Thielecke [37] and later Hasegawa [30, Prop. 1].)

Outline. The section is structured as follows. In §3.1 we give a definition of concurrent 2-monads adapted to the strict 2-categorical setting of this paper. Then in §3.2 we introduce concurrent dialogue 2-categories, axiomatized with fork and join. Finally §3.3 and §3.4 we look separately at call-by-name and call-by-value concurrent evaluation. Note that in this section we only discuss abstract categorical notions. We will mention concrete models in §4.

3.1 Concurrent 2-monads

We first give our notion of concurrent 2-monad, adapted from [35] and [33]. To avoid oversized diagrams, we denote tensor products by juxtaposition, and applications by subscripts, e.g. T^2_{AB} means $T(T(A \otimes B))$; and we sometimes omit subscripts for transformation components.

▶ **Definition 18.** *A* concurrent 2-monad *on a symmetric monoidal 2-category $(\mathcal{C}, \otimes, I)$ is a 2-monad (T, η, μ) on $\mathcal{C}$ equipped with a 2-natural transformation $\chi_{A,B} : TA \otimes TB \to T(A \otimes B)$ and a modification*

$$\begin{array}{ccccc} TTA \otimes TTB & \xrightarrow{\chi_{TA,TB}} & T(TA \otimes TB) & \xrightarrow{T\chi_{A,B}} & TT(A \otimes B) \\ \downarrow{\scriptstyle \mu_A \otimes \mu_B} & & \Downarrow \kappa_{A,B} & & \downarrow{\scriptstyle \mu_{A,B}} \\ TA \otimes TB & & \xrightarrow[\chi_{A,B}]{} & & T(A \otimes B) \end{array}$$

subject to the following axioms:

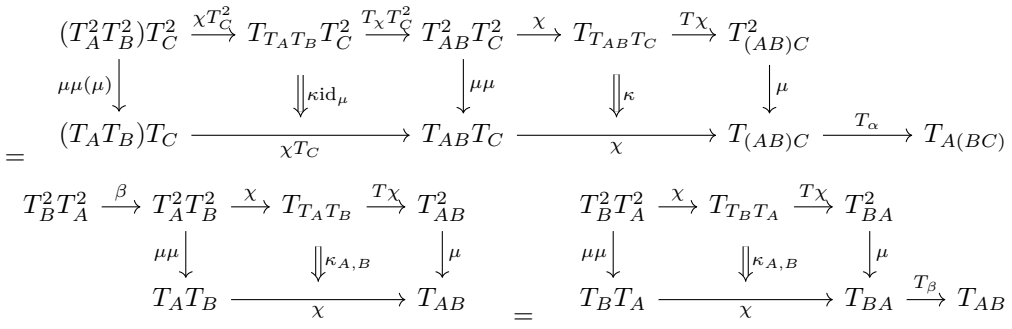
- **Weakly symmetric monoidal 2-monad:** *The tuple (T, η_I, χ) defines a symmetric monoidal 2-functor; the 2-natural transformation η is monoidal, meaning that the diagram*

$$\begin{array}{ccc} AB & \xrightarrow{\eta_A \eta_B} & T_A T_B \\ & \searrow^{\eta_{AB}} & \downarrow{\scriptstyle \chi_{A,B}} \\ & & T_{AB} \end{array}$$

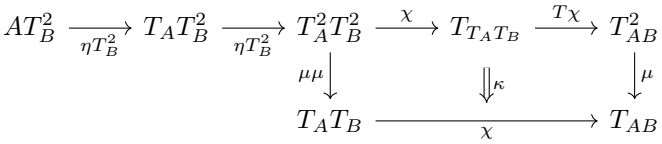
commutes; and the 2-natural transformation μ, equipped with κ, is symmetric monoidal in a lax sense specified by the following equality of 2-cells[2]*:*

$$\begin{array}{ccccccccc} (T^2_A T^2_B)T^2_C & \xrightarrow{\alpha} & T^2_A(T^2_B T^2_C) & \xrightarrow{T^2_A \chi} & T^2_A T_{T_B T_C} & \xrightarrow{T^2_A T_\chi} & T^2_A T^2_{BC} & \xrightarrow{\chi} & T_{T_A T_{BC}} & \xrightarrow{T_\chi} & T^2_{A(BC)} \\ & & \downarrow{\scriptstyle \mu(\mu\mu)} & & \Downarrow \mathrm{id}_\mu \kappa & & \downarrow{\scriptstyle \mu\mu} & & \Downarrow \kappa & & \downarrow{\scriptstyle \mu} \\ & & T_A(T_B T_C) & & \xrightarrow[T_A \chi]{} & & T_A(T_{BC}) & \xrightarrow[\chi]{} & & & T_{A(BC)} \end{array}$$

[2] At first sight it looks like our 2-cell equations are not well typed. This is because (for unfortunate space reasons) we have only included the non-identity parts of the pasting diagrams. In other words, our diagrams do type-check, because the 1-cell boundaries are equal, but this equality must be reconstructed using diagrams of 1-cells.



- **Hiding:** *The whiskered 2-cell below is an identity 2-cell.*



▶ Remark 19. Compared to the general bicategorical theory in [33], the only novelties are the strict 2-categorical setting and the symmetry of the monoidal structure. Note that it is standard for monoidal monads that the monoidal unit $I \to TI$ should coincide with the monad unit at I. Lax-monoidal transformations must also be compatible with monoidal units, but here it is automatic.

The *hiding* axiom in the definition has a different nature. Intuitively, composing with an effect-free program in parallel or sequentially should make no difference. More formally, the purpose is to ensure that the composite $A \otimes TB \xrightarrow{\eta_A \otimes TB} TA \otimes TB \xrightarrow{\chi_{A,B}} T(A \otimes B)$ satisfies the axioms for a strength. "Hiding" is game semantics terminology: precomposing with $\eta \circ \eta$ hides the effectful part in the left-hand component of the game $T_A^2 \otimes T_B^2$.

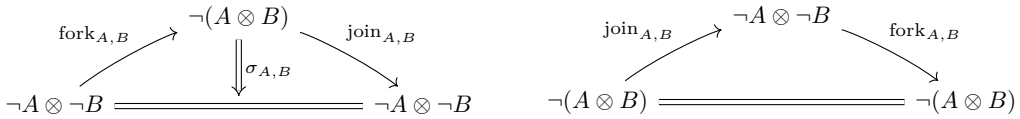
3.2 Concurrent dialogue 2-categories

We now state the central definition of this paper.

▶ **Definition 20** (Concurrent dialogue 2-category)**.** *A* concurrent dialogue 2-category *is a dialogue 2-category* $(\mathcal{C}, \otimes, \neg)$ *with the additional data of 2-natural transformations*

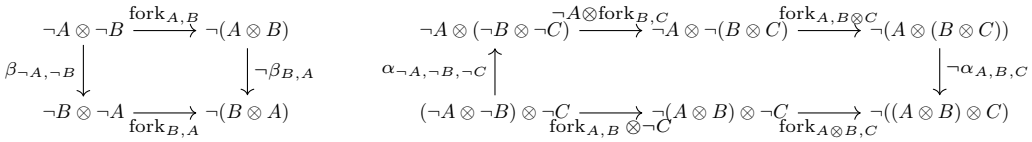
$$\mathrm{fork}_{A,B} : \neg A \otimes \neg B \longrightarrow \neg(A \otimes B) \qquad \mathrm{join}_{A,B} : \neg(A \otimes B) \longrightarrow \neg A \otimes \neg B$$

and a modification $\sigma : \mathrm{join} \circ \mathrm{fork} \to \mathrm{id}$*, such that* $\mathrm{fork} \circ \mathrm{join} = \mathrm{id}$*, as per the diagrams below.*

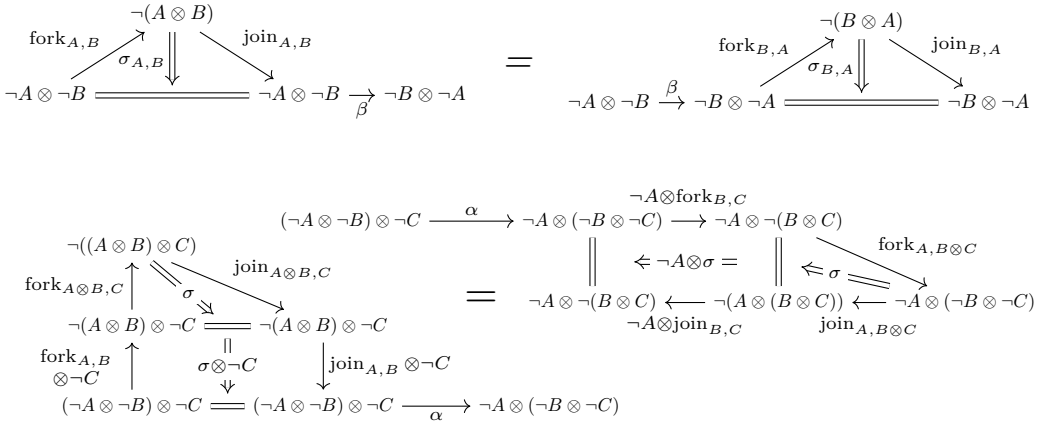


This data is subject to additional axioms given below.

- **Associativity and braiding:** *Writing* α *and* β *for the associator and braiding in the symmetric monoidal structure of* $\mathcal{C}$*, the diagrams*

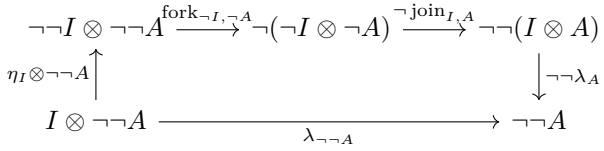


commute, as well as the analogous two diagrams for join*, and two coherence laws hold for components of* σ*:*



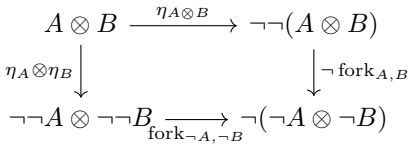
For these coherence laws, we provide full pasting diagrams in Appendix A.

- **Compatibility with monoidal units:** *The following diagram commutes:*



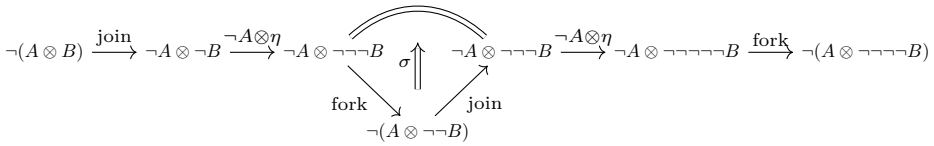
The corresponding diagram for the the right unitor ρ *holds as a consequence. Note that* fork *and* join *themselves have no unit, so this axiom involves both transformations together.*

- **Compatibility of fork and negation:** *The following diagram commutes:*



This is only required for fork*, whose domain has a top-level tensor. No such axiom can be expressed for* join *(nor is it needed).*

- **Hiding:** *The following whiskered 2-cell should be an identity:*



This definition has a clear intuitive meaning in game semantics (detailed later in §4.1), and convenient abstract properties we discuss below in this section. We also note the following useful consequence of the compatibility of fork and negation:

▶ **Lemma 21.** *In a concurrent dialogue 2-category, the two maps*

$$A \otimes \neg B \xrightarrow{\eta_A \otimes \neg B} \neg\neg A \otimes \neg B \xrightarrow{\mathrm{fork}_{\neg A, B}} \neg(\neg A \otimes B) \qquad \neg A \otimes B \xrightarrow{\neg A \otimes \eta_B} \neg A \otimes \neg\neg B \xrightarrow{\mathrm{fork}_{A, \neg B}} \neg(A \otimes \neg B)$$

correspond to each other under the bijection $\Phi_{A \otimes \neg B, \neg A \otimes B}$ *induced by the adjunction.*

3.3 Call-by-value semantics with concurrent evaluation

We look at applications of this concurrent structure. Our most significant result is:

▶ **Theorem 22.** *In a concurrent dialogue 2-category, the continuation 2-monad $\neg\neg$ is a concurrent 2-monad, with $\chi_{A,B} = \neg\neg A \otimes \neg\neg B \xrightarrow{\text{fork}} \neg(\neg A \otimes \neg B) \xrightarrow{\neg\,\text{join}} \neg\neg(A \otimes B)$. In particular, it admits a strength $A \otimes \neg\neg B \xrightarrow{\eta_A \otimes \neg\neg B} \neg\neg A \otimes \neg\neg B \xrightarrow{\chi_{A,B}} \neg\neg(A \otimes B)$.*

▶ Remark 23. The continuation 2-monad $\neg\neg$ already has a canonical strength (Lemma 10), and nothing seems to imply that the two should coincide, since monads can have several strengths. But it is expected that in concrete examples they *will* be the same (e.g. §2.4).

The proof of Theorem 22 involves constructing the modification κ, and verifying all axioms. This is mostly routine, but we give some details in Appendix B.

We can use this structure to design a variation on the usual call-by-value semantics for our linear λ-calculus (§2.3). The construction is straightforward: the semantics of types is exactly the same, and the semantics of terms is also the same except for the transformation named $(\blacktriangleleft\blacktriangleright)$, which we replace by the transformation χ. Write $[\![t]\!]^v_{\|}$ for the semantics of a term t in this model.

▶ **Proposition 24.** *For types A and B, any concurrent dialogue 2-category has a 2-cell*

$$\neg\neg\neg([\![A]\!]^v \otimes \neg[\![B]\!]^v) \otimes \neg\neg[\![A]\!]^v \;\; \underset{\chi}{\overset{\blacktriangleleft\blacktriangleright}{\Downarrow}} \;\; \neg\neg(\neg([\![A]\!]^v \otimes \neg[\![B]\!]^v) \otimes [\![A]\!]^v)$$

where $\blacktriangleleft\blacktriangleright$ is any of the two possible sequential evaluation orders induced by the strength of $\neg\neg$. By compositionality of the semantics, we get a 2-cell $[\![t]\!]^v_{\|} \to [\![t]\!]^v$ for every term t.

The meaning of this 2-cell will depend on the model: it might be used to compare convergence, possible returned values, probabilities, etc. in the two semantics.

3.4 Call-by-name semantics with concurrent evaluation

To illustrate that these combinators can be applied beyond call-by-value, we sketch a parallel semantics in which, whenever an evaluation term $(\lambda x.t)\ s$ is encountered, both function and argument are evaluated in parallel. unlike in call-by-value, if the function returns without a call to x, then the whole program can return immediately.

This semantics follows the structure of the call-by-name semantics in §2.3. The only difference is that we use a new evaluation 1-cell $\text{ev}^{\|}$, defined only for CBN types, using that they are all interpreted as tensor products of negated objects (clear by induction).

▶ **Definition 25** (Parallel evaluation for CBN types)**.** *In a dialogue 2-category $(\mathcal{C}, \otimes, I, \bot)$, consider objects A and B with $A = \neg A_1 \otimes \cdots \otimes \neg A_n$. Write $\text{fork}_A : \bigotimes_{i=1}^n \neg\neg A_i \to \neg A$ for the n-ary fork, evjoin_A for the 1-cell*

$$I \cong \bigotimes_{i=1}^n I \xrightarrow{\bigotimes_i^n \eta_I} \bigotimes_{i=1}^n \neg\neg I \xrightarrow{\bigotimes_i \neg\text{ev}_{A_i,I}} \bigotimes_{i=1}^n \neg(\neg A_i \otimes A_i) \xrightarrow{\bigotimes_{i=1}^n \text{join}} \bigotimes_{i=1}^n \neg\neg A_i \otimes \neg A_i$$

and finally define $\mathrm{ev}^{\|}_{\neg A,B}$ *as*

$$\neg(\neg A \otimes B) \otimes \neg A \xrightarrow{\mathrm{fork}_{\neg A \otimes B, A}} \neg(\neg A \otimes B \otimes A) \xrightarrow{\neg(\mathrm{fork}_A \otimes B \otimes A)} \neg(\bigotimes_{i=1}^{n} \neg\neg A_i \otimes B \otimes A)$$
$$\xrightarrow{\cong} \neg(B \otimes \bigotimes_{i=1}^{n} \neg\neg A_i \otimes \neg A_i) \xrightarrow{\neg(B \otimes \mathrm{evjoin}_A)} \neg(B \otimes I) \xrightarrow{\cong} \neg B.$$

Using this alternative evaluation 1-cell (and everything else kept the same) gives a new semantics of terms, which we write $[\![t]\!]^n_{\|}$.

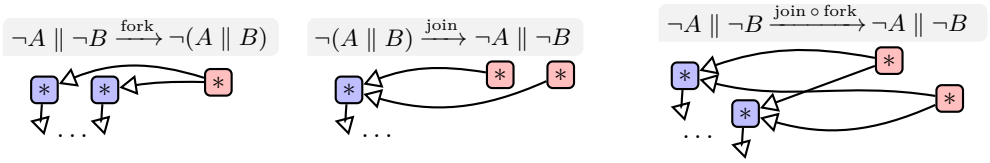
▶ **Proposition 26.** *For A and B as in Definition 25, there is a 2-cell $\mathrm{ev}_{\neg A,B} \longrightarrow \mathrm{ev}^{\|}_{\neg A,B}$, and thus a 2-cell $[\![t]\!]^n \longrightarrow [\![t]\!]^n_{\|}$ for every term t of our language.*

Although we save a full account for further work, this combinatorial presentation recovers existing constructions given directly in concurrent games on event structures [5, 6].

4 Further illustrations and degenerate models

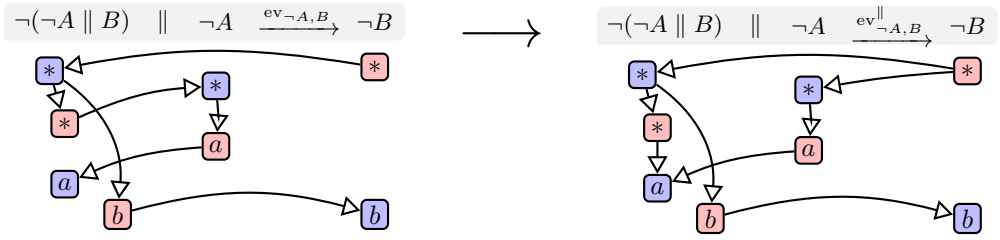
4.1 Game semantics

We return to our simple game model, in which the fork and join combinators are particularly clear. Recall from §2.4 that for a game A, $\neg A$ is the pomset $A^{\perp}$ with an appended negative initial element $*$. The dialogue 2-category $\mathcal{G}$ admits a concurrent structure, drawn below. (The "…" indicate the point at which a strategy behaves like an identity on $A \parallel B$.)



On the right above is the composite join ∘ fork, which should be contrasted with the identity strategy on $\neg A \parallel \neg B$: the latter would have two "parallel arrows" in the partial order. A 2-cell σ with the required properties can easily be defined, essentially as an instance of Proposition 2.

▶ **Example 27.** We contrast the strategy $\mathrm{ev}_{\neg A,B}$ with the parallel variant $\mathrm{ev}^{\|}_{\neg A,B}$ from Definition 25, in the case $A = B = \neg I$. Let a and b denote the unique moves of A and B.



4.2 Degenerate models in dialogue 1-categories

Every category can be regarded as a *locally discrete* 2-category, i.e. with only identity 2-cells. This way every dialogue category is also a dialogue 2-category, and we can look for concurrent structures there.

But locally discrete concurrent dialogue 2-categories are necessarily degenerate: σ must be the identity, and so $\neg A \otimes \neg B$ and $\neg(A \otimes B)$ are isomorphic objects. The derived 2-cell κ associated with the continuation 2-monad is also the identity. In that case the concurrent 2-monad is just a monoidal monad on the underlying category, and it is well known that monoidal monads are the same as commutative monads [20]. But continuation 2-monads are commutative if and only if they are idempotent – a drastic restriction for modelling program effects.[3] Thus the expressivity of concurrent dialogue 1-categories is very limited, and only covers models in which $\neg(A \otimes B) \cong \neg A \otimes \neg B$ and $A \mapsto \neg\neg A$ is a closure operator.

4.3 Degenerate models in cartesian closed 2-categories

The structures we introduce appear to be of limited applicability when the monoidal product $\otimes$ coincides with a cartesian product $\times$ in the underlying 2-category $\mathcal{C}$. In that case, the type of $\mathrm{join}_{A,B}$ forces it to be the pairing of two functions $\neg(A \times B) \to \neg A$ and $\neg(A \times B) \to \neg B$, which goes against the informal intuition that the output of join should be a pair of "entangled" continuations. A full no-go theorem is left for further work, but we show a limitation in the case of a cartesian closed 2-category with an initial object (e.g. **Cat**).

▶ **Proposition 28.** *Let $\mathcal{C}$ be a cartesian closed 2-category with an initial object 0, and consider the dialogue structure induced by any object $R \in \mathcal{C}$ (so $\neg A = A \Rightarrow R$). If $\mathcal{C}$ has concurrent structure, then R must be a terminal object.*

Proof. We instantiate the naturality of the transformation join at 1-cells $! : 0 \to A$ and $\mathrm{id}_B : B \to B$. Since $\mathcal{C}$ is cartesian closed, we have $0 \times B \cong 0$, and so $0 \times B \Rightarrow R \cong 1$. We deduce that $\mathrm{join}_{A,B}$ must be equal to

$$(A \times B) \Rightarrow R \xrightarrow{!} 1 \xrightarrow{\langle \mathrm{join}_{A,0}, \mathrm{join}_{0,B} \rangle} (A \Rightarrow R) \times (B \Rightarrow R)$$

for all A and B. Taking $A = B = 1$ we then obtain that the morphism $! : R \to 1$ has a retraction $1 \xrightarrow{\langle \mathrm{join}_{1,0}, \mathrm{join}_{0,1} \rangle} R \times R \xrightarrow{\mathrm{fork}_{1,1}} R$. So $R \cong 1$. ◀

4.4 Syntactic illustration in a CPS language

Finally we look at what our combinators may look like in a typed target language for CPS transformations. In what follows we call *source language* the linear λ-calculus of the previous section. The *target language* has types and terms given by

$$A, B ::= I \mid \bot \mid A \otimes B \mid \neg A$$
$$s, t ::= () \mid x \mid \lambda x.t \mid s\ t \mid s \otimes t \mid \mathrm{let}\ x \otimes y = s\ \mathrm{in}\ t \mid \mathrm{join}\, t \mid \mathrm{fork}\, t$$

with typing rules given below. We also use syntactic sugar $\lambda(x \otimes y).t := \lambda p.(\mathrm{let}\ x \otimes y = p\ \mathrm{in}\ t)$.

$$\frac{\Gamma, x : A \vdash t : \bot}{\Gamma \vdash \lambda x.t : \neg A} \qquad \frac{\Gamma \vdash s : \neg A \qquad \Delta \vdash t : A}{\Gamma, \Delta \vdash s\ t : \bot} \qquad \frac{}{\vdash () : I} \qquad \frac{\Gamma \vdash s : A \qquad \Delta \vdash t : B}{\Gamma, \Delta \vdash s \otimes t : A \otimes B}$$

$$\frac{\Gamma \vdash s : A \otimes B \qquad \Delta, x : A, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathrm{let}\ x \otimes y = s\ \mathrm{in}\ t : C} \qquad \frac{\Gamma \vdash t : \neg A \otimes \neg B}{\Gamma \vdash \mathrm{fork}\, t : \neg(A \otimes B)} \qquad \frac{\Gamma \vdash t : \neg(A \otimes B)}{\Gamma \vdash \mathrm{join}\, t : \neg A \otimes \neg B}$$

[3] Note that, by itself, the commutativity of a 2-monad does not mean that all concurrent structures on it are uninteresting [33, Example 6.2]. This is only true for continuation monads, because commutativity implies idempotence.

It is well-established that the core of this language (without fork and join) is an appropriate target language for call-by-value and call-by-name CPS transformations. With the new primitives we can look for parallel variants of CPS. As an example, we contrast the standard call-by-value translation (arbitrarily fixing a left-first evaluation) with a parallel version. The translations are respectively denoted $(\!|-|\!)^v$ and $(\!|-|\!)^v_{||}$. Types are translated inductively as $(\!|I|\!)^v = (\!|I|\!)^v_{||} = I$ and $(\!|A \multimap B|\!)^v = (\!|A \multimap B|\!)^v_{||} = \neg((\!|A|\!)^v \otimes \neg(\!|B|\!)^v)$. A term $\Gamma \vdash t : A$ is translated as $(\!|\Gamma|\!)^v \vdash (\!|t|\!)^v : \neg\neg(\!|A|\!)^v$ (and the same for $(\!|-|\!)^v_{||}$), as described in the table:

Source t	Standard CPS $(\!|t|\!)^v$	Parallel CPS $(\!|t|\!)^v_{||}$
$()$	$\lambda k.k\ ()$	$\lambda k.k\ ()$
x	$\lambda k.k\ x$	$\lambda k.k\ x$
$\lambda x.t$	$\lambda k.k(\lambda(x \otimes h).(\!|t|\!)^v h)$	$\lambda k.k(\lambda(x \otimes h).(\!|t|\!)^v_{||}\ h)$
$s\ t$	$\lambda k.(\!|s|\!)^v(\lambda f.(\!|t|\!)^v(\lambda x.k\ (f\ x)))$	$\lambda k.\,\mathrm{fork}((\!|s|\!)^v_{||} \otimes (\!|t|\!)^v_{||})(\mathrm{join}(\lambda(u \otimes a).u(a \otimes k)))$

For a language with side-effects and a proper 2-dimensional structure on terms (consider for instance the partial order induced by may-termination in a non-deterministic language, or the 2-dimensional syntax of [11]), our concurrent dialogue 2-categories will be a basis for soundness and adequacy theorems, with compositional reasoning principles at this level.

5 Conclusion: related and further work

We have given a categorical axiomatization of concurrent continuation models based on dialogue 2-categories extended with join and fork combinators. The 2-categorical structure is essential for this axiomatization. It has a clear operational meaning in game semantics and we have also showcased other applications.

Instead of dialogue categories we could have used an alternative notion of continuation model (e.g. [37, 17, 36]). Relationships between these models are fairly well understood, and we expect that principles for concurrency can be transferred.

We emphasize that the non-cartesian tensor exploited here is not ultimately a barrier to the interpretation of non-linear languages. Indeed, with an appropriate resource modality on a dialogue category, one recovers a semantic model for full call-by-push-value with continuations [8]. We expect that this will not interfere with the concurrent combinators.

On the side of concrete models, we note that game semantics and concurrency already have a significant shared history, including e.g. [12, 9, 28]. This appears to be the first time that parallel combinators are given a categorical axiomatization, and game models for parallel execution should all fit in this framework. An idea in this direction was already suggested by Melliès [27, p. 243].

In this short paper our focus has been on motivation and illustration of the 2-categorical setting. In further work we will study concurrency in non-linear settings with sum types and recursion. Our axiomatization will provide a solid foundation. Another avenue is in connection with the use of parallel evaluation in normalization proofs for the λ-calculus [2]. Finally, we will investigate more general 2-categorical models for concurrent effectful programming, including call-by-value and call-by-name but going beyond the special case of continuation models.

References

1 John C Baez and Kenny Courser. Structured cospans. *Theory & Applications of Categories*, 35, 2020.

2 H P Barendregt. The lambda calculus, its syntax and semantics. *Studies in Logic and the Foundations of Mathematics*, 103, 1984.

3 Josh Berdine, Peter W. O'Hearn, Uday S. Reddy, and Hayo Thielecke. Linear continuation-passing. *High. Order Symb. Comput.*, 15(2-3):181–208, 2002. doi:10.1023/A:1020891112409.

4 Simon Castellan, Pierre Clairambault, Silvain Rideau, and Glynn Winskel. Games and strategies as event structures. *Logical Methods in Computer Science*, 13, 2017. doi:10.23638/LMCS-13(3:35)2017.

5 Simon Castellan, Pierre Clairambault, and Glynn Winskel. The parallel intensionally fully abstract games model of PCF. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 232–243. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.31.

6 Simon Castellan, Pierre Clairambault, and Glynn Winskel. Observably deterministic concurrent strategies and intensional full abstraction for parallel-or. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK*, volume 84 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.FSCD.2017.12.

7 Pierre Clairambault, Federico Olimpieri, and Hugo Paquet. From thin concurrent games to generalized species of structures. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*, pages 1–14. IEEE, 2023. doi:10.1109/LICS56636.2023.10175681.

8 Pierre-Louis Curien, Marcelo P. Fiore, and Guillaume Munch-Maccagnoni. A theory of effects and resources: adjunction models and polarised calculi. In Rastislav Bodík and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 44–56. ACM, 2016. doi:10.1145/2837614.2837652.

9 Claudia Faggian and Mauro Piccolo. Partial orders, event structures and linear strategies. In *International Conference on Typed Lambda Calculi and Applications*, pages 95–111. Springer, 2009. doi:10.1007/978-3-642-02273-9_9.

10 Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glynn Winskel. The cartesian closed bicategory of generalised species of structures. *Journal of the London Mathematical Society*, 77(1):203–220, 2008.

11 Marcelo Fiore and Philip Saville. A type theory for cartesian closed bicategories. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785708.

12 Dan R Ghica and Andrzej S Murawski. Angelic semantics of fine-grained concurrency. *Annals of Pure and Applied Logic*, 151(2-3):89–114, 2008. doi:10.1016/j.apal.2007.10.005.

13 Timothy G Griffin. A formulae-as-type notion of control. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47–58, 1989. doi:10.1145/96709.96714.

14 Masahito Hasegawa. Semantics of linear continuation-passing in call-by-name. In Yukiyoshi Kameyama and Peter J. Stuckey, editors, *Functional and Logic Programming, 7th International Symposium, FLOPS 2004, Nara, Japan, April 7-9, 2004, Proceedings*, volume 2998 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2004. doi:10.1007/978-3-540-24754-8_17.

15 Tony Hoare. Unifying semantics for concurrent programming. In Bob Coecke, Luke Ong, and Prakash Panangaden, editors, *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky - Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, volume 7860 of *Lecture Notes in Computer Science*, pages 139–149. Springer, 2013. doi:10.1007/978-3-642-38164-5_10.

16 Tony Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent kleene algebra and its foundations. *J. Log. Algebraic Methods Program.*, 80(6):266–296, 2011. doi:10.1016/j.jlap.2011.04.005.

17 Martin Hofmann and Thomas Streicher. Completeness of continuation models for lambda-mu-calculus. *Inf. Comput.*, 179(2):332–355, 2002. doi:10.1006/inco.2001.2947.

18 Niles Johnson and Donald Ying Yau. *2-Dimensional Categories*. Oxford university press, Oxford, 2021.

19 Axel Kerinec, Giulio Manzonetto, and Federico Olimpieri. Why are proofs relevant in proof-relevant models? *Proc. ACM Program. Lang.*, 7(POPL):218–248, 2023. doi:10.1145/3571201.

20 Anders Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23:113–120, 1972.

21 Stephen Lack. A 2-categories companion. In *Towards higher categories*, pages 105–191. Springer, 2009.

22 Paul Blain Levy. Adjunction models for call-by-push-value with stacks. In Richard Blute and Peter Selinger, editors, *Category Theory and Computer Science, CTCS 2002, Ottawa, Canada, August 15-17, 2002*, volume 69 of *Electronic Notes in Theoretical Computer Science*, pages 248–271. Elsevier, 2002. doi:10.1016/S1571-0661(04)80568-1.

23 Paul Blain Levy. Pointer games. *Call-By-Push-Value: A Functional/Imperative Synthesis*, pages 169–203, 2003.

24 Paul-André Melliès. Asynchronous games 3 an innocent model of linear logic. In Lars Birkedal, editor, *Proceedings of the 10th Conference on Category Theory in Computer Science, CTCS 2004, Copenhagen, Denmark, August 12-14, 2004*, volume 122 of *Electronic Notes in Theoretical Computer Science*, pages 171–192. Elsevier, 2004. doi:10.1016/j.entcs.2004.06.057.

25 Paul-André Mellies. Categorical semantics of linear logic. *Panoramas et syntheses*, 27:15–215, 2009.

26 Paul-André Melliès. Game semantics in string diagrams. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 481–490. IEEE Computer Society, 2012. doi:10.1109/LICS.2012.58.

27 Paul-André Melliès. Une étude micrologique de la négation. *Habilitation à diriger des recherches. Université Paris VII*, page 19, 2017.

28 Paul-André Melliès. Categorical combinatorics of scheduling and synchronization in game semantics. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019. doi:10.1145/3290336.

29 Paul-André Melliès and Léo Stefanesco. Concurrent separation logic meets template games. *CoRR*, abs/2005.04453, 2020. doi:10.48550/arXiv.2005.04453.

30 Paul-André Melliès and Nicolas Tabareau. Resource modalities in tensor logic. *Ann. Pure Appl. Log.*, 161(5):632–653, 2010. doi:10.1016/j.apal.2009.07.018.

31 Eugenio Moggi. Notions of computation and monads. *Information and computation*, 93(1):55–92, 1991. doi:10.1016/0890-5401(91)90052-4.

32 Hugo Paquet. *Probabilistic concurrent game semantics*. PhD thesis, University of Cambridge, Computer Laboratory, 2020. doi:10.17863/CAM.61919.

33 Hugo Paquet and Philip Saville. Effectful semantics in bicategories: strong, commutative, and concurrent pseudomonads. In Pawel Sobocinski, Ugo Dal Lago, and Javier Esparza, editors, *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024*, pages 61:1–61:15. ACM, 2024. doi:10.1145/3661814.3662130.

34 Silvain Rideau and Glynn Winskel. Concurrent strategies. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 409–418. IEEE Computer Society, 2011. doi:10.1109/LICS.2011.13.

35 Exequiel Rivas and Mauro Jaskelioff. Monads with merging, 2019. URL: https://inria.hal.science/hal-02150199/document.

36 Peter Selinger. Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Mathematical structures in computer science*, 11(2):207–260, 2001. URL: `http://journals.cambridge.org/action/displayAbstract?aid=68983`.

37 Hayo Thielecke. *Categorical structure of continuation passing style*. PhD thesis, University of Edinburgh. College of Science and Engineering, 1997.

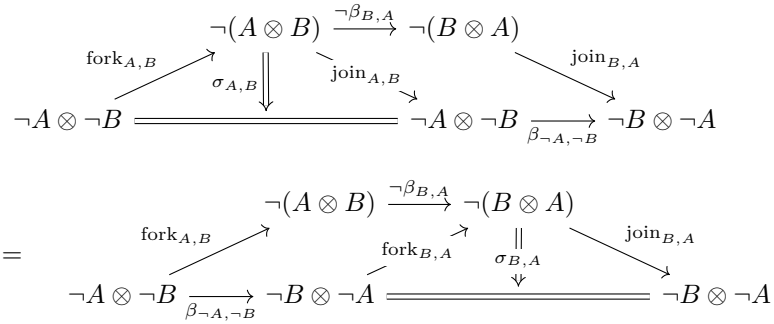
38 Philip Wadler. Call-by-value is dual to call-by-name. In Colin Runciman and Olin Shivers, editors, *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25-29, 2003*, pages 189–201. ACM, 2003. `doi:10.1145/944705.944723`.

39 Glynn Winskel and Mogens Nielsen. Models for concurrency. In *Handbook of logic in computer science (vol. 4) semantic modelling*, pages 1–148. 1995.

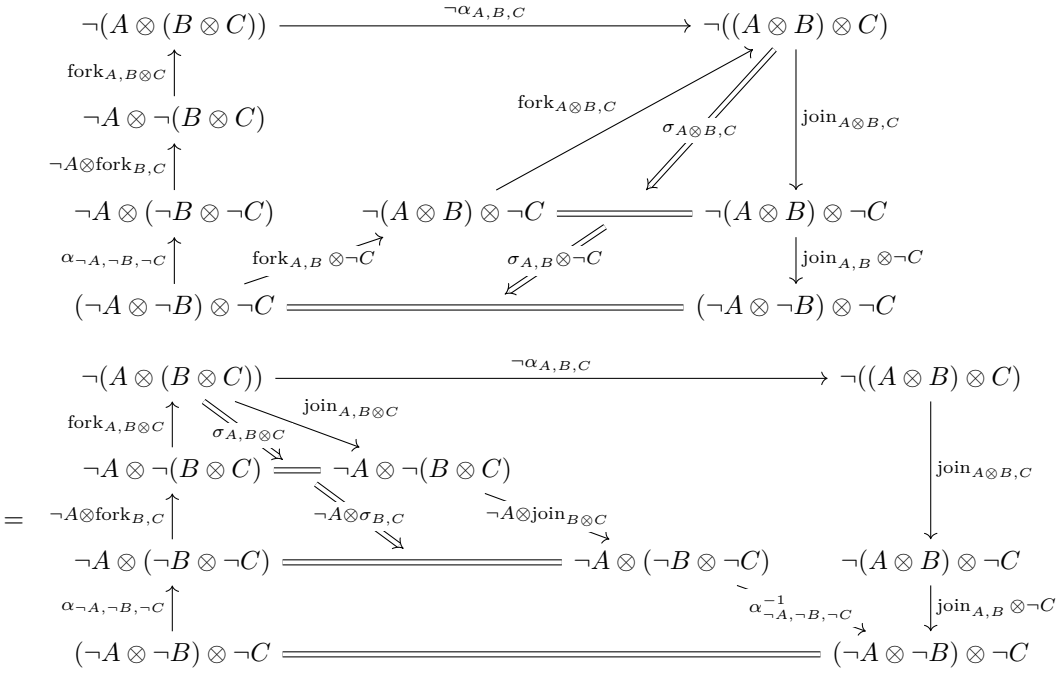
A Full coherence laws for σ in a concurrent dialogue 2-categories

The two coherence laws for components of σ in Definition 20 are given below.

Braiding.

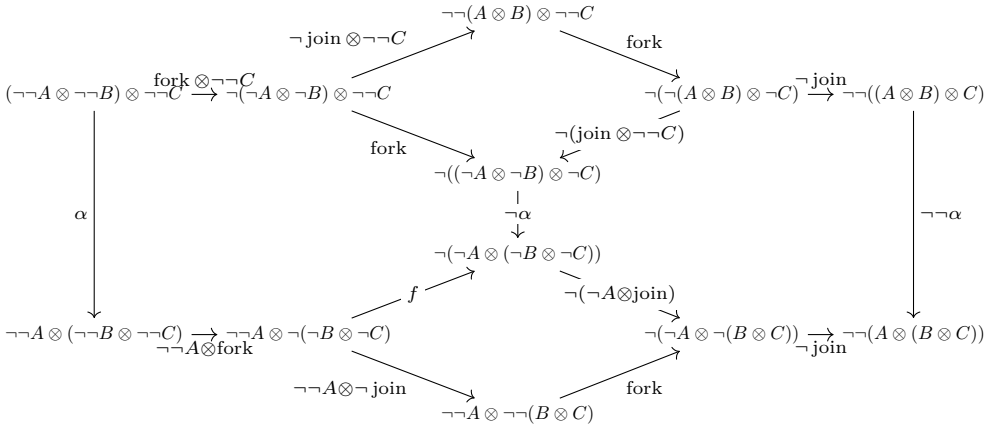


Associativity.



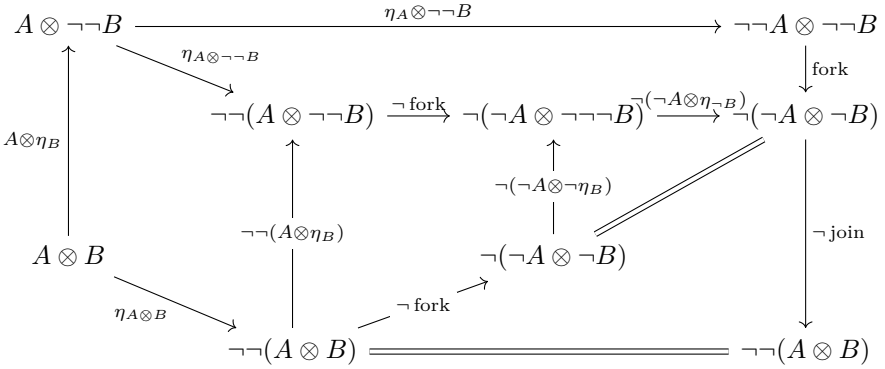
B Proof notes: the continuation monad $\neg\neg$ is concurrent

Symmetric monoidal functor. The following diagram commutes by associativity of join and fork, and by naturality:

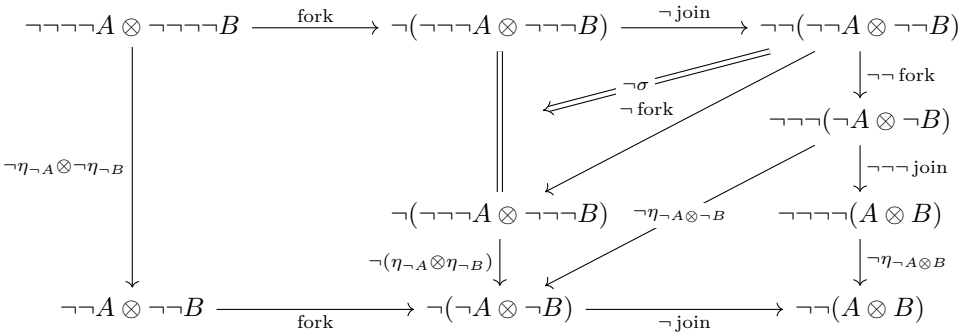


The left unit diagram commutes by definition and the right unit diagram below commutes by symmetry.

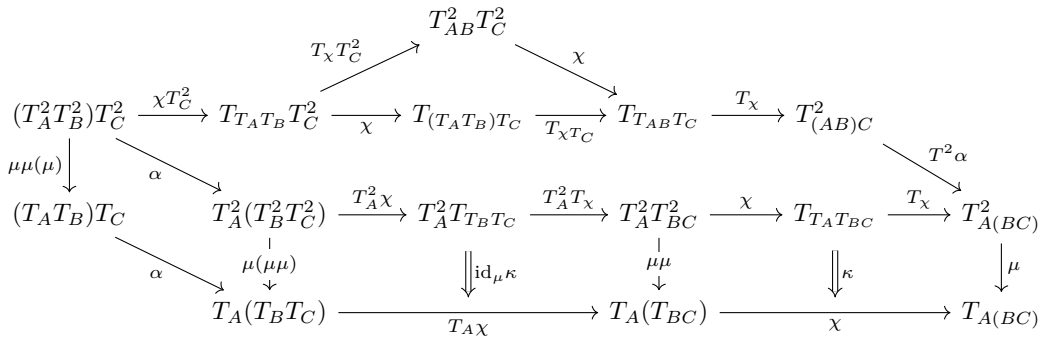
The 2-natural transformation η is monoidal. The proof amounts to the diagram below, in which the top subdiagram uses the compatibility of negation with fork.



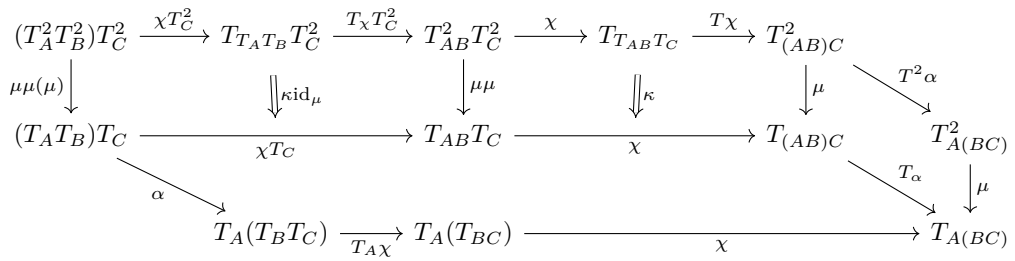
The 2-natural transformation μ is lax-monoidal. The modification component $\kappa_{A,B}$ is given by



The proof that this κ is coherent involves a tedious sequence of rewriting steps on 2-cells. The key steps uses the coherence axioms on σ. For illustration we include one of the coherence axioms for κ in full form:



=



Hiding. The hiding condition for concurrent 2-monads is derived from the hiding condition for concurrent dialogue 2-categories, by rearranging the pasting diagram for the whiskering

$$\kappa \circ (\eta_{\neg\neg A} \otimes \mathrm{id}_{\neg\neg\neg\neg B}) \circ (\eta_A \otimes \mathrm{id}_{\neg\neg\neg\neg B})$$

in several steps. We omit the details for reasons of space.