Composition and Merging of Assume-Guarantee Contracts Are Tensor Products

Inigo Incer

University of Michigan, Ann Arbor, MI 48109 USA
iir@umich.edu

Abstract. We show that the operations of composition and merging of contracts are part of the tensor product structure of the algebra of contracts.

1 Introduction

In the 2000s, Benveniste et al. [2] proposed contracts to provide semantics to the component-based design methodologies common in the system industries. Instead of using detailed mathematical models to represent components, contract-based design can yield insight about a system using the assume-guarantee specifications of its components. The basic ingredients of analysis are the contracts and the algebraic operations on these objects.

The algebraic operations on contracts were discovered piecemeal. The original paper on the topic [2] discusses conjunction, disjunction, and composition. Conjunction, which is the greatest lower bound induced from the contract order, was proposed as a way to bring various aspects, dubbed *viewpoints*, into the same contract. Disjunction was induced from the order on contracts as the least upper bound; its use in system design was not known for several years, until it was pointed out that its role is in product lines—see [4], page 21. This shows that logical considerations can sometimes be ahead of our practical understanding. Composition was defined as the operation that yields the specification of a system from the specifications of components being interconnected. The composition formula was constructed from intuitive considerations and is reminiscent of the composition axiom proposed by M. Abadi and L. Lamport in the early 90s [1]. A fourth binary operation, merging, was proposed later [9] on axiomatic grounds to fuse into a single contract multiple viewpoints that we want to hold simultaneously for a component. This shows that intuition can also be ahead of theory.

For each of the four basic operations just described, a new operation was discovered that optimally undoes it. These are dubbed *adjoint* operations per their role in category theory—see Section 6.8 of [4]. For instance, composition builds a system from its constituent elements; its adjoint, quotient, finds the optimal specifications of objects that need to be added to a system to make it satisfy a goal. The following diagram provided in [4] shows how the various contract operations relate to each other.

$$\begin{tikzcd}
 & \text{Conjunction } \wedge \arrow[r, Rightarrow, "\text{Right adjoint}"] \arrow[d, Leftrightarrow, "\text{Dual}"'] & \text{Implication } \rightarrow \arrow[d, Leftrightarrow, "\text{Dual}"] & & \text{Composition } \parallel \arrow[r, Rightarrow, "\text{Right adjoint}"] \arrow[d, Leftrightarrow, "\text{Dual}"'] & \text{Quotient } / \arrow[d, Leftrightarrow, "\text{Dual}"] \\
\text{Order} \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & & & \text{Axiom} \arrow[ur, Rightarrow] \arrow[dr, Rightarrow] & & \\
 & \text{Disjunction } \vee \arrow[r, Rightarrow, "\text{Left adjoint}"] & \text{Coimplication } \nrightarrow & & \text{Merging } \bullet \arrow[r, Rightarrow, "\text{Left adjoint}"] & \text{Separation } \div
\end{tikzcd}$$

This diagram shows that two operations are obtained from the notion of order, and two by axiom: composition and merging. The operations of conjunction, disjunction, and its adjoints provide contracts with significant algebraic structure [5]. Given the practical, widespread use of merging and composition, we wonder whether these operations have an algebraic interpretation, as well.

Contributions. The purpose of this note is to study the algebraic roles of the operations of merging and composition in the theory of contracts. It is shown that these operations can be understood as tensor products. This interpretation enables the identification of these operations without an appeal to axiom.

Outline. Section 2 contains basic definitions about contracts. This paper makes no contributions in this section. Section 3 investigates the tensor product of contracts. Except where explicitly indicated, this section contains new material. We discuss these results in Section 4.

2 Assume-guarantee contracts

To define contracts, we assume we have access to a language used to express properties. We will require this language to have Boolean semantics (i.e., Boolean connectives plus the excluded middle), and as a result, we shall refer to it as $\mathbf{B}$. The notation in this section follows closely [4] and [6].

Definition 1. *Given a Boolean algebra* $\mathbf{B}$*, its contract algebra* $\mathbb{C}(\mathbf{B})$ *is given by*

$$\mathbb{C}(\mathbf{B}) = \{(a, g) \in \mathbf{B}^{\text{op}} \times \mathbf{B} \mid a \vee g = 1_{\mathbf{B}}\}.$$

Some observations about this definition:

- A contract is a pair (a, g) of *assumptions* and *guarantees.* The semantics of a contract, and the way the contract operations were originally defined, was by understanding a contract as a specification that requires a component to satisfy the contract's guarantees g when the environment in which it operates satisfies the contract's assumptions a.
- The condition $a \vee g = 1_{\mathbf{B}}$ in the above definition is there to give to the pairs (a, g) the semantics of assume-guarantee reasoning. Note that if the assumptions a do not hold, g must be asserted; in other words, this condition says that if the assumptions do not hold, the guarantees hold ipso facto.
- $\mathbf{B}^{\text{op}}$ means the opposite of $\mathbf{B}$, the algebra in which $\wedge$ has been switched with $\vee$, and the top with the bottom elements. This notation is there to embed in the definition (1) of the contract algebra of $\mathbf{B}$ the following partial order:

$$(a, g) \leq (a', g') \text{ iff } g \leq g' \text{ and } a' \leq a.$$

Observe this definition matches intuition. It says that a contract (a, g) is stricter—or a *refinement*—of contract (a', g') if the guarantees of the stricter contract are stricter than those of the more relaxed contract, and the assumptions the stricter contract makes on its environment are more relaxed than those that the more relaxed contract makes.

$\mathbb{C}(\mathbf{B})$ is a bounded partial order:

Corollary 1. *The contracts* $0 \coloneqq (1_{\mathbf{B}}, 0_{\mathbf{B}})$ *and* $1 \coloneqq (0_{\mathbf{B}}, 1_{\mathbf{B}})$ *are, respectively, the bottom and top of* $\mathbb{C}(\mathbf{B})$.

2.1 Conjunction and disjunction

The definition of contracts, with its embedded notion of order, immediately generates two binary operations: conjunction and disjunction, defined as the greatest lower bound and least upper bound for arbitrary pairs of contracts, respectively. Note that these two operations come perforce from the definition of contracts.

Proposition 1. *Let* $(a, g), (a', g') \in \mathbb{C}(\mathbf{B})$. *The greatest lower bound (or conjunction) and the least upper bound (or disjunction) of these contracts are, respectively, given by*

$$(a, g) \wedge (a', g') = (a \vee a', g \wedge g') \text{ and } (a, g) \vee (a', g') = (a \wedge a', g \vee g').$$

Proof. $(a'', g'') \leq (a, g)$ and $(a'', g'') \leq (a', g')$ means that $g'' \leq g$ and $a \leq a''$ and $g'' \leq g'$ and $a' \leq a''$. This is equivalent to $g'' \leq g \wedge g'$ and $a \vee a' \leq a''$, which means that $(a'', g'') \leq (a \vee a', g \wedge g')$. This proves the expression for conjunction. A similar procedure leads to the formula for disjunction. □

2.2 Axioms for composition and merging

Two types of models of $\mathbf{B}$ are associated with contracts over $\mathbf{B}$. They go by the names of environments and implementations.

Definition 2. *An* environment *of a contract* $(a, g) \in \mathbb{C}(\mathbf{B})$ *is a model of* a, *and an* implementation *of this contract is a model of* g.

Given contracts $\mathcal{C}$ and $\mathcal{C}'$, two operations are defined by axiom:

Axiom 1. The *merger* of contracts $\mathcal{C}$ and $\mathcal{C}'$, denoted $\mathcal{C} \bullet \mathcal{C}'$, is the most relaxed contract $\mathcal{C}_m$ satisfying these two requirements:

1. An implementation of $\mathcal{C}_m$ is an implementation of both $\mathcal{C}$ and $\mathcal{C}'$.
2. An environment of $\mathcal{C}$ that is also an environment of $\mathcal{C}'$ is an environment of $\mathcal{C}_m$.

To define the axiom for the composition of contracts, we assume that models of $\mathbf{B}$ have a well-defined composition operation. For details, we refer the reader to Chapter 5 of [3] and [7].

Axiom 2. The *composition* of contracts $\mathcal{C}$ and $\mathcal{C}'$, denoted $\mathcal{C} \parallel \mathcal{C}'$, is the most refined contract $\mathcal{C}_c$ satisfying these two requirements:

1. The composition of an implementation of $\mathcal{C}$ with an implementation of $\mathcal{C}'$ yields an implementation of $\mathcal{C}_c$.
2. The composition of an environment of $\mathcal{C}_c$ with an implementation of one of the contracts being composed is an environment of the other contract being composed.

The application of these two axioms leads to the following closed form expressions for composition and merging:

Corollary 2. *Given $(a, g), (a', g') \in \mathbb{C}(\mathbf{B})$, the closed form expressions to compute merging and composition are*

$$(a, g) \bullet (a', g') = (a \wedge a', (a \wedge a') \to (g \wedge g')) \quad \textit{and} \tag{1}$$

$$(a, g) \parallel (a', g') = ((g \wedge g') \to (a \wedge a'), g \wedge g')\,. \tag{2}$$

Here, $\to$ stands for the implication operator of Boolean algebras. Observe the difference in the way we arrived at the expressions to compute conjunction/disjunction and composition/merging. The former were obtained by algebraic means, i.e., as the LUB and GLB of the refinement order, while the latter were obtained by axiom. The purpose of this note is to understand the role that merging and composition play in the algebra of contracts. In particular, we would like to understand whether their closed form expressions can be obtained by algebraic means. It turns out they are related to tensor products.

3 Tensor products of contracts

In this section, we study a notion of tensor product of contracts with the tools so far developed. First, we discuss the notion of a tensor product over bimodules of monoids. Then we apply these notions to contracts.

3.1 Tensor products of bimodules over monoids

Tensor products are normally defined over ring modules. We will need to modify standard definitions slightly to obtain tensor products over structures with actions of monoids. A comprehensive reference on tensor products is Chapter 11 of [10].

We recall that a monoid $(B, \cdot, 1_B)$ is a semigroup $(B, \cdot)$ with an identity element 1_B. Let $(M, +, 0_M)$ be a commutative monoid, i.e., a monoid with a commutative operation $+$. We say that M is a B-bimodule if it is equipped with operations $\cdot \colon M, B \to M$ and $\cdot \colon B, M \to M$ satisfying the following properties for all $x, y \in B$ and $m, m' \in M$:

- $1_B \cdot m = m = m \cdot 1_B$
- $(xy) \cdot m = x \cdot (y \cdot m)$

- $m \cdot (xy) = (m \cdot x) \cdot y$
- $(x \cdot m) \cdot y = x \cdot (m \cdot y)$
- $x \cdot (m + m') \cdot y = (x \cdot m \cdot y) + (x \cdot m' \cdot y)$

The operation that applies an element of B on the left is called a *left action*, and the operation that applies it on the right a *right action*.

Suppose that M, N, and O are B-bimodules. A B-linear map $f\colon M \to O$ is a map satisfying the following identities for all $m, m' \in M$ and $x, y \in B$:

- $f(m + m') = f(m) + f(m')$
- $x \cdot f(m) \cdot y = f(x \cdot m \cdot y)$

A B-bilinear map $f\colon M, N \to O$ is a map satisfying the following identities for all $m, m' \in M$, $n, n' \in N$, and $x, y \in B$:

- $f(m + m', n) = f(m, n) + f(m', n)$
- $f(m, n + n') = f(m, n) + f(m, n')$
- $x \cdot f(m, n) \cdot y = f(x \cdot m \cdot y, n) = f(m, x \cdot n \cdot y)$

A tensor product of B-bimodules M and N is a B-bimodule $M \otimes N$ together with a B-bilinear map $\tau\colon M, N \to M \otimes N$ satisfying the following property: for every B-bilinear map $f\colon M, N \to O$, there is a unique B-linear map $M \otimes N \to O$ making the following diagram commute:

$$\begin{tikzcd} M, N \arrow[r, "f"] \arrow[d, "\tau"] & O \\ M \otimes N \arrow[ur, dashed] & \end{tikzcd}$$

3.2 The tensor product of conjunctive monoids of contracts

Let **Bool** be the category of Boolean algebras, and **Mon** that of monoids. We consider the functors $\mathbf{M}_\wedge, \mathbb{C}^M_\wedge \colon \mathbf{Bool} \to \mathbf{Mon}$ defined as follows:

$$\mathbf{B} \xmapsto{\mathbf{M}_\wedge} (|\mathbf{B}|, \wedge, 1_\mathbf{B}) \qquad f \xmapsto{\mathbf{M}_\wedge} f \quad \text{and}$$
$$\mathbf{B} \xmapsto{\mathbb{C}^M_\wedge} (|\mathbb{C}(\mathbf{B})|, \wedge, 1) \qquad f \xmapsto{\mathbb{C}^M_\wedge} f \times f,$$

where $\mathbf{B}$ and f are, respectively, an object and a morphism in **Bool**. In other words, $\mathbf{M}_\wedge(\mathbf{B})$ is the monoid obtained by only remembering the conjunction operation and the distinguished element $1_\mathbf{B}$ of the Boolean algebra $\mathbf{B}$; similarly, $\mathbb{C}^M_\wedge(\mathbf{B})$ is the monoid obtained by only remembering the conjunction operation and the distinguished element 1 of the contract algebra $\mathbb{C}(\mathbf{B})$—see Corollary 1. Contract monoids are studied in Section 6.10.1 of [4]. For ease of syntax, throughout this paper we will say $\mathbf{B}$-bimodules and $\mathbf{B}$-linear/bilinear maps instead of $\mathbf{M}_\wedge(\mathbf{B})$-bimodules, etc.

We now turn the monoid $\mathbb{C}^M_\wedge(\mathbf{B})$ into a $\mathbf{B}$-bimodule through the following operations:

$$x \cdot (a, g) \coloneqq ((x \wedge a), x \to g) \qquad \text{and}$$

$$(a,g)\cdot x := (a, a \to (x \wedge g)) \qquad\qquad (x \in \mathbf{B}, (a,g) \in \mathbb{C}(\mathbf{B}))\,.$$

This left action was defined in Section 6.11 of [4]. The right action is, to the best of our knowledge, new. First, we observe that $(x \wedge a) \vee (x \to g) = 1_{\mathbf{B}} = a \vee (a \to (x \wedge g))$. This means that $x \cdot (a,g)$ and $(a,g)\cdot x$ are elements of $\mathbb{C}(\mathbf{B})$. Second, the operations just defined have the following interpretation: given a contract $\mathcal{C} \in \mathbb{C}(\mathbf{B})$, the left operation $x \cdot \mathcal{C}$ has the effect of adding to the contract $\mathcal{C}$ an additional assumption x. Similarly, the contract $\mathcal{C}\cdot x$ has the same assumptions as $\mathcal{C}$, but it has additional guarantees x. In other words, $x \cdot \mathcal{C}$ adds x to the assumptions of $\mathcal{C}$, and $\mathcal{C}\cdot x$ adds x to its guarantees.

Proposition 2. *Under the left and right operations just defined, $\mathbb{C}^M_\wedge(\mathbf{B})$ is a $\mathbf{B}$-bimodule.*

Proof. Let $x, y \in \mathbf{B}$ and $(a,g),(a',g') \in \mathbb{C}(\mathbf{B})$. We verify the conditions of a bimodule stated in Section 3.2:

- $1_{\mathbf{B}}\cdot(a,g) = ((1_{\mathbf{B}} \wedge a), 1_{\mathbf{B}} \to g) = (a,g) = (a, a \to (1_{\mathbf{B}} \wedge g)) = (a,g)\cdot 1_{\mathbf{B}}$.
- $(x \wedge y)\cdot(a,g) = x \cdot (y \cdot (a,g))$, as show in the proof of Proposition 6.11.1 of [4].
- $$\begin{aligned}(a,g)\cdot(x\wedge y) &= (a, a \to (g \wedge (x \wedge y)))\\ &= (a, a \to ((g\wedge x)\wedge y))\\ &= (a, a \to (g \wedge x))\cdot y\\ &= ((a,g)\cdot x)\cdot y.\end{aligned}$$
- $$\begin{aligned}(x\cdot(a,g))\cdot y &= ((x\wedge a), x \to g)\cdot y\\ &= ((x \wedge a), (x\wedge a) \to (y \wedge (x \to g)))\\ &= ((x\wedge a), (x \wedge a) \to (y \wedge g))\\ &= ((x \wedge a), x \to (a \to (y \wedge g)))\\ &= x \cdot (a, a \to (y \wedge g))\\ &= x \cdot ((a,g)\cdot y)\,.\end{aligned}$$
- We will use the following identity:
$$(a \vee a') \to (g \wedge g') = g \wedge g' = (a \to g) \wedge (a' \to g'). \tag{3}$$

  We can now verify the property:
$$\begin{aligned} x\cdot ((a,g)\wedge(a',g'))\cdot y &= x \cdot ((a \vee a', g \wedge g'))\cdot y\\ &= x\cdot(a\vee a', (a \vee a') \to (y \wedge g \wedge g'))\\ &= ((x\wedge a)\vee(x\wedge a'), x \to ((a\vee a') \to (y \wedge g \wedge g')))\\ &\overset{(3)}{=} \left((x\wedge a)\vee(x\wedge a'), x \to \begin{pmatrix}(a \to (y \wedge g))\wedge\\ (a' \to (y\wedge g'))\end{pmatrix}\right)\\ &= \left((x\wedge a)\vee(x\wedge a'), \begin{pmatrix}((x\wedge a) \to (y\wedge g))\wedge\\ ((x\wedge a')\to(y\wedge g'))\end{pmatrix}\right)\\ &= (x\wedge a, (x\wedge a)\to(y\wedge g))\wedge(x\wedge a', (x\wedge a')\to(y\wedge g'))\\ &= (x\cdot(a,g)\cdot y)\wedge(x\cdot(a',g')\cdot y).\end{aligned}$$

□

Theorem 1. $\mathbb{C}^M_\wedge(\mathbf{B})$ *together with the map* $(\cdot) \bullet (\cdot) \colon \mathbb{C}^M_\wedge(\mathbf{B}), \mathbb{C}^M_\wedge(\mathbf{B}) \to \mathbb{C}^M_\wedge(\mathbf{B})$ *given by* (1) *is the tensor product* $\mathbb{C}^M_\wedge(\mathbf{B}) \otimes \mathbb{C}^M_\wedge(\mathbf{B})$*.*

Proof. We know from Proposition 2 that $\mathbb{C}^M_\wedge(\mathbf{B})$ is a $\mathbf{B}$-bimodule. First we verify that $(\cdot) \bullet (\cdot)$ is a $\mathbf{B}$-bilinear map. Observe that this operation is commutative, so we only have to check additivity/linearity for one argument. We carry out the following verifications:

– Let $\mathcal{C}, \mathcal{C}', \mathcal{C}'' \in \mathbb{C}^M_\wedge(\mathbf{B})$. Then $(\mathcal{C} \wedge \mathcal{C}') \bullet \mathcal{C}'' = (\mathcal{C} \bullet \mathcal{C}'') \wedge (\mathcal{C}' \bullet \mathcal{C}'')$ from Proposition 6.10.4 of [4].
– Let $x, y \in \mathbf{B}$ and $(a, g), (a', g') \in \mathbb{C}^M_\wedge(\mathbf{B})$. We have

$$\begin{aligned} x \cdot ((a,g) \bullet (a',g')) \cdot y &= (x \wedge a \wedge a', (x \wedge a \wedge a') \to (y \wedge g \wedge g')) \\ &= (x \wedge a, (x \wedge a) \to (y \wedge g)) \bullet (a', g') \\ &= (x \cdot (a,g) \cdot y) \bullet (a', g'). \end{aligned}$$

We have shown that merging is a $\mathbf{B}$-bilinear map $\mathbb{C}^M_\wedge(\mathbf{B}), \mathbb{C}^M_\wedge(\mathbf{B}) \to \mathbb{C}^M_\wedge(\mathbf{B})$. Now suppose N is a $\mathbf{B}$-bimodule and that $f \colon \mathbb{C}^M_\wedge(\mathbf{B}), \mathbb{C}^M_\wedge(\mathbf{B}) \to N$ is a $\mathbf{B}$-bilinear map. Let $e \coloneqq (1_\mathbf{B}, 1_\mathbf{B}) \in \mathbb{C}^M_\wedge(\mathbf{B})$. Define the map $\hat{f} \colon \mathbb{C}^M_\wedge(\mathbf{B}) \to \mathbb{C}^M_\wedge(\mathbf{B})$ by

$$\hat{f}(\mathcal{C}) = f(e, \mathcal{C}).$$

Since f is $\mathbf{B}$-bilinear, $\hat{f}$ is $\mathbf{B}$-linear. Let $(a, g), (a', g') \in \mathbb{C}^M_\wedge(\mathbf{B})$. We observe that

$$\begin{aligned} f((a,g),(a',g')) &= f(a \cdot e \cdot g, (a',g')) = f(e, a \cdot (a',g') \cdot g) \\ &= f(e, (a \wedge a', (a \wedge a') \to (g \wedge g'))) = f(e, (a,g) \bullet (a',g')) \\ &= \hat{f}((a,g) \bullet (a',g')). \end{aligned}$$

This means that there exists a $\mathbf{B}$-linear map $\hat{f}$ making the following diagram commute:

$$\begin{tikzcd} \mathbb{C}^M_\wedge(\mathbf{B}), \mathbb{C}^M_\wedge(\mathbf{B}) \arrow[r, "f"] \arrow[d, "(\cdot)\bullet(\cdot)"] & N \\ \mathbb{C}^M_\wedge(\mathbf{B}) \arrow[ur, "\hat{f}"'] & \end{tikzcd} \tag{4}$$

Suppose there is a second $\mathbf{B}$-linear map $h \colon \mathbb{C}^M_\wedge(\mathbf{B}) \to \mathbb{C}^M_\wedge(\mathbf{B})$ such that

$$\hat{f}(\mathcal{C} \bullet \mathcal{C}') = h(\mathcal{C} \bullet \mathcal{C}')$$

for all $\mathcal{C}, \mathcal{C}' \in \mathbb{C}^M_\wedge(\mathbf{B})$. By setting $\mathcal{C}' = e$ in this expression, we obtain that $\hat{f} = h$. It follows that there is a unique map $\hat{f}$ making (4) commute. The statement of the theorem follows. □

3.3 The tensor product of disjunctive monoids of contracts

In this section, we consider the disjunctive contract monoid functor $\mathbb{C}^M_\vee : \mathbf{Bool} \to \mathbf{Mon}$ (see Proposition 6.10.1 of [4]) defined as follows:

$$\mathbf{B} \overset{\mathbb{C}^M_\vee}{\longmapsto} (|\mathbb{C}(\mathbf{B})|, \vee, 0) \qquad\qquad f \overset{\mathbb{C}^M_\vee}{\longmapsto} f \times f.$$

Proposition 6.10.2 of [4] shows that $\mathbb{C}^M_\vee(\mathbf{B})$ and $\mathbb{C}^M_\wedge(\mathbf{B})$ are isomorphic *as monoids* via the map $(\cdot)^{-1} \colon \mathbb{C}(\mathbf{B}) \to \mathbb{C}(\mathbf{B})$ defined by

$$(a, g)^{-1} = (g, a).$$

Let $\mathcal{C} = (a, g) \in \mathbb{C}^M_\vee(\mathbf{B})$ and $\mathcal{C}' = (g, a) \in \mathbb{C}^M_\wedge(\mathbf{B})$. There are unique left and right actions of $\mathbf{M}_\wedge(\mathbf{B})$ on $\mathbb{C}^M_\vee(\mathbf{B})$ that make $(\cdot)^{-1}$ a linear map:

$$\begin{aligned}
x \cdot (a, g) = x \cdot \mathcal{C} = x \cdot (\mathcal{C}')^{-1} &\coloneqq (x \cdot \mathcal{C}')^{-1} = (x \cdot (g, a))^{-1} \\
&= (x \wedge g, x \to a)^{-1} = (x \to a, x \wedge g) \\
(a, g) \cdot x = \mathcal{C} \cdot x = (\mathcal{C}')^{-1} \cdot x &\coloneqq (\mathcal{C}' \cdot x)^{-1} = ((g, a) \cdot x)^{-1} \\
&= (g, g \to (x \wedge a))^{-1} = (g \to (x \wedge a), g).
\end{aligned}$$

This means that $(\cdot)^{-1}$ is an isomorphism of $\mathbb{C}^M_\vee(\mathbf{B})$ and $\mathbb{C}^M_\wedge(\mathbf{B})$ *as* $\mathbf{B}$*-bimodules*. Now we can prove our main result.

Theorem 2. *$\mathbb{C}^M_\vee(\mathbf{B})$ together with the map $(\cdot) \parallel (\cdot) \colon \mathbb{C}^M_\vee(\mathbf{B}), \mathbb{C}^M_\vee(\mathbf{B}) \to \mathbb{C}^M_\vee(\mathbf{B})$ given by* (2) *is the tensor product $\mathbb{C}^M_\vee(\mathbf{B}) \otimes \mathbb{C}^M_\vee(\mathbf{B})$.*

Proof. Consider the $\mathbf{B}$-bimodule N and the $\mathbf{B}$-bilinear map $f \colon \mathbb{C}^M_\vee(\mathbf{B}), \mathbb{C}^M_\vee(\mathbf{B}) \to N$. Since $(\cdot)^{-1}$ is an isomorphism, for every such map f, there exist a unique $\mathbf{B}$-bilinear map $\tilde{f} \colon \mathbb{C}^M_\wedge(\mathbf{B}), \mathbb{C}^M_\wedge(\mathbf{B}) \to N$ such that

$$f(\mathcal{C}, \mathcal{C}') = \tilde{f}\left(\mathcal{C}^{-1}, (\mathcal{C}')^{-1}\right).$$

Per Proposition 2, there is a unique $\mathbf{B}$-linear map $\hat{f} \colon \mathbb{C}^M_\wedge(\mathbf{B}) \to N$ such that

$$\tilde{f}(\mathcal{C}, \mathcal{C}') = \hat{f}(\mathcal{C} \bullet \mathcal{C}').$$

Per the isomorphism $(\cdot)^{-1}$, there is a unique $\mathbf{B}$-linear map $\check{f} \colon \mathbb{C}^M_\vee(\mathbf{B}) \to N$ such that

$$\hat{f} = \check{f}(\cdot)^{-1}.$$

Putting this reasoning together, we have shown that for every $\mathbf{B}$-bilinear map f, there is a unique $\mathbf{B}$-linear map $\check{f}$ such that

$$f(\mathcal{C}, \mathcal{C}') = \check{f}\left(\left(\mathcal{C}^{-1} \bullet (\mathcal{C}')^{-1}\right)^{-1}\right) = \check{f}\left(\mathcal{C} \parallel \mathcal{C}'\right),$$

where the second equality is due to the duality of merging and composition—see Section 6.4 of [4]. In other words, for every **B**-bilinear map f, there is a unique **B**-linear map $\breve{f}$ making the following diagram commute:

$$\begin{tikzcd}
\mathbb{C}^M_\vee(\mathbf{B}), \mathbb{C}^M_\vee(\mathbf{B}) \arrow[r, "f"] \arrow[d, "(\cdot)\|(\cdot)"] & N \\
\mathbb{C}^M_\vee(\mathbf{B}) \arrow[ur, "\breve{f}"'] &
\end{tikzcd},$$

which proves the theorem. □

4 Discussion and concluding remarks

We have shown that the operations of merging and composition of contracts, which up to now have been defined by axiom, belong naturally in the tensor product structure of contracts. The diagram we presented in Section 1—showing all known binary contract operations—can now be recast as shown below. The diagonals of the bottom square correspond to the preheap[1] identities described in Section 6.9 of [4].

$$\begin{tikzcd}
 & \text{Implication } \rightarrow \arrow[r, Leftrightarrow, "\text{Dual}"] & \text{Coimplication } \nrightarrow \\
\text{Order } \Rightarrow & \text{Conjunction } \wedge \arrow[u, Rightarrow, "\text{Right adjoint}"] \arrow[r, Leftrightarrow, "\text{Dual}"] \arrow[d, Rightarrow, "\text{Tensor product}"] & \text{Disjunction } \vee \arrow[u, Rightarrow, "\text{Left adjoint}"'] \arrow[d, Rightarrow, "\text{Tensor product}"'] \\
 & \text{Merging } \bullet \arrow[r, Leftrightarrow, "\text{Dual}"] \arrow[d, Rightarrow, "\text{Left adjoint}"'] \arrow[dr, Leftrightarrow, "\text{Preheap}"] & \text{Composition } \| \arrow[d, Rightarrow, "\text{Right adjoint}"] \arrow[dl, Leftrightarrow, "\text{Preheap}"'] \\
 & \text{Separation } \div \arrow[r, Leftrightarrow, "\text{Dual}"'] & \text{Quotient } /
\end{tikzcd}$$

Acknowledgements

We thank the reviewers of our paper for their helpful comments. This work was supported by ASEE and NSF through the eFellows postdoctoral program. During the completion of this work, the author was with the California Institute of Technology, Pasadena, CA, USA.

References

1. Abadi, M., Lamport, L.: Composing specifications. ACM Transactions on Programming Languages and Systems **15**(1), 73–132 (January 1993)

[1] Preordered heaps, or preheaps, are studied in [8].

2. Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., Sofronis, C.: Multiple viewpoint contract-based specification and design. In: Formal Methods for Components and Objects, 6^{th} International Symposium (FMCO 2007), Lecture Notes in Computer Science, vol. 5382, pp. 200–225. Springer Verlag, Berlin Heidelberg (2008). https://doi.org/10.1007/978-3-540-92188-2
3. Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J.B., Reinkemeier, P., Sangiovanni-Vincentelli, A.L., Damm, W., Henzinger, T.A., Larsen, K.G.: Contracts for system design. Foundations and Trends® in Electronic Design Automation **12**(2-3), 124–400 (2018). https://doi.org/10.1561/1000000053
4. Incer, I.: The Algebra of Contracts. Ph.D. thesis, EECS Department, University of California, Berkeley (May 2022), `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-99.html`
5. Incer, I.: An adjunction between Boolean algebras and a subcategory of Stone algebras. arXiv:2309.04135 (2023)
6. Incer, I., Benveniste, A., Sangiovanni-Vincentelli, A.: Some algebraic aspects of assume-guarantee reasoning. arXiv:2309.08875 (2023)
7. Incer, I., Benveniste, A., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Hypercontracts. In: Deshmukh, J., Havelund, K., Perez, I. (eds.) NASA Formal Methods. pp. 674–692. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-06773-0_36
8. Incer, I., Mangeruca, L., Villa, T., Sangiovanni-Vincentelli, A.L.: The quotient in preorder theories. In: Raskin, J.F., Bresolin, D. (eds.) Proceedings 11th International Symposium on Games, Automata, Logics, and Formal Verification, Brussels, Belgium, September 21-22, 2020. Electronic Proceedings in Theoretical Computer Science, vol. 326, pp. 216–233. Open Publishing Association, Brussels, Belgium (2020). https://doi.org/10.4204/EPTCS.326.14
9. Passerone, R., Incer, I., Sangiovanni-Vincentelli, A.L.: Coherent extension, composition, and merging operators in contract models for system design. ACM Trans. Embed. Comput. Syst. **18**(5s) (Oct 2019). https://doi.org/10.1145/3358216
10. Wodzicki, M.: Notes on Category Theory. Lecture notes for Math H113—UC Berkeley (Nov 2016)