Theory and Applications of Categories, Vol. 41, No. 57, 2024, pp. 2041–2057.

AN ADJUNCTION BETWEEN BOOLEAN ALGEBRAS AND A SUBCATEGORY OF STONE ALGEBRAS

INIGO INCER

Abstract. We consider Stone algebras with a distinguished element e satisfying the identity $e \to x = \neg\neg x$ for all elements x of the algebra. We provide an adjunction between the category of such algebras and that of Boolean algebras. This adjunction turns out to involve the concept of assume-guarantee contracts, which has numerous applications throughout engineering and computer science.

1. Introduction

The algebra of contracts [Benveniste et al., 2018, Incer Romeo, 2022] has been an object of attention in computer science and engineering due to its capability to support compositional design.

1.1. Definition. *Given a Boolean algebra B, its contract algebra $\mathbf{C}(B)$ has elements $(a, g) \in B^{op} \times B$ such that $a \vee g = 1_B$. The contract algebra $\mathbf{C}(B)$ is a lattice with operations*

$$\begin{aligned}(a,g) \wedge (a',g') &= (a \vee a', g \wedge g'),\\ (a,g) \vee (a',g') &= (a \wedge a', g \vee g'), \text{ and}\\ (a',g') \to (a,g) &= ((a \wedge \neg a') \vee (g' \wedge \neg g), \neg g' \vee g).\end{aligned} \tag{1}$$

The top and bottom elements of this lattice are, respectively, $1 = (0_B, 1_B)$ and $0 = (1_B, 0_B)$. Given a Boolean algebra morphism f, we let $\mathbf{C}f = f \times f$. We will refer to $\mathbf{C}$ as the *contract functor* (its target category will be introduced in due time).

As contracts are increasingly used in engineering and computer science[1], we are motivated to understand more precisely the nature of contract algebras. Such is the purpose of this paper: to understand what makes the contract functor unique. We will introduce a subcategory of Stone algebras and will show that the contract functor is the left adjoint of the functor that maps these algebras to their closures. This defines the contract functor in terms of a universal property.

The author is grateful to Professor George Bergman for helpful comments on an earlier version of this manuscript and to the anonymous reviewers for suggesting many improvements. This work was supported by NSF and ASEE through an eFellows postdoctoral fellowship. The author was with the California Institute of Technology in Pasadena, CA, during the completion of this work.

Received by the editors 2023-09-09 and, in final form, 2024-12-03.

Transmitted by Valeria de Paiva. Published on 2024-12-09.

2020 Mathematics Subject Classification: 08A70, 08C05.

Key words and phrases: Stone algebras, contracts, assume-guarantee reasoning.

[1] See [Benveniste et al., 2018] and the numerous papers citing it across engineering disciplines.

This paper is an extended version of the preprint [Incer, 2023] and is organized as follows. Section 2 contains preliminary information about assume-guarantee contracts. We introduce augmented Stone algebras and discuss some of their properties in Section 3. Section 4 proves that the contract functor is part of an adjunction between Boolean algebras and augmented Stone algebras. Section 5 discusses implications and potential extensions of this work.

2. Preliminaries on assume-guarantee contracts

The complexity of large-scale systems engineering has motivated the development of compositional theoretical frameworks to support system design. One such frameworks is the theory of assume-guarantee contracts, proposed by Benveniste et al. [Benveniste et al., 2008], which formalizes—and was inspired in—several design methodologies commonly practiced in industry. It is typical in the practice of engineering to develop large systems by interconnecting components developed by third parties. These components often come with a datasheet that states what the component provides and under what expectations on its context of operation a component may be relied upon to deliver its guarantees. For example, a processor's datasheet will state guarantees on the performance of its communication peripherals and assumptions on temperature ranges, input voltages, oscillator frequencies, etc., under which said performance is guaranteed. In other words, component datasheets come in assume-guarantee form.

One problem with the design methodologies applied in systems industries today (e.g., aerospace, automotive, etc.) is that specifications are expressed informally in natural languages, limiting the analysis that can be effected on them. Assume-guarantee contracts were introduced to enable a design process whereby multiple parties develop components independently, and analysis carried out in advance can guarantee that the interconnection of these components is safe and yields a system with desired properties. This methodology has the potential to alleviate frictions in supply chains, as integrators and suppliers can now exchange, in addition to mutually-binding legal contracts, a technical contract that ensures the integrator that the system will meet its objectives if the supplier generates a correct implementation of the technical contract.

2.1. Original formulation. Contracts were originally defined using the behavioral approach to system modeling. Behavioral modeling, with its origins in both computer science and control systems[2], is the idea that a component in a design should be represented by the behaviors we can witness from it. If the components are software routines, they would be represented as sets of execution traces. If the components are electromechanical, they would be represented as sets of the solutions to the equations that describe their dynamics. Thus, behavioral modeling implies the existence of a universe of behaviors $\mathcal{B}$. A component in the system is defined as a subset of $\mathcal{B}$.

[2]A brief overview of behavioral modeling can be found in Section 2.3 of [Incer Romeo, 2022].

In behavioral modeling, a property is also defined as a set of behaviors. In contrast to components, whose elements are the behaviors that we can witness from them, properties (also called trace properties) contain behaviors that we approve of because they are deemed safe or secure or because they mean our design has reached its destination. Given a component M and a property P, we say that M satisfies P, denoted $M \models P$, if $M \subseteq P$, i.e., if every behavior of M is a behavior that we approve of (because it is safe, secure, performant, etc.).

Behavioral modeling is compositional. Given components M and M'. The system obtained by interconnecting these two objects is given by

$$M \parallel M' \stackrel{\text{def}}{=} M \cap M'.$$

One way to interpret this is by thinking of the behaviors of the system $M \parallel M'$ as those that satisfy the constraints imposed by both M and M'.

Now that we have the notions of components and properties, we can discuss contracts. The purpose of a contract is to attach an assume-guarantee specification to a component. We want to state what the component guarantees and under what assumptions on its context of operation it can deliver these guarantees. Thus, in the original definition [Benveniste et al., 2008], a contract is a pair (A, G), where $A, G \subseteq \mathcal{B}$ are trace properties over $\mathcal{B}$ that represent, respectively, the assumptions and guarantees of the contract.

Contracts make predicates over components. A component E is said to be an environment for the contract (A, G) if $E \models A$, i.e., if E satisfies the assumptions of the contract. A component M is said to be an implementation of the contract if $\forall E \subseteq A.\ M \parallel E \models G$, i.e., if M provides the guarantees of the contract when operating in an environment of the contract. Observe that this last statement simply means that $M \subseteq G \cup \neg A$, i.e., component M is required to satisfy its guarantees G only when the assumptions A are satisfied.

The semantics of assume-guarantee reasoning are embedded in the definitions of environments and implementations. We say that contracts (A, G) and (A', G') are equivalent if they have the same environments and the same implementations. This means these contracts are equivalent when

$$A = A' \quad \text{and} \quad G \cup \neg A = G' \cup \neg A'.$$

In particular, a contract (A, G) is equivalent to $(A, G \cup \neg A)$. The latter contract has the largest set of guarantees that a contract equivalent to (A, G) can possibly have. These contracts are said to be in *saturated* or *canonical* form. Observe that a contract (A'', G'') is in saturated form if $G'' = G'' \cup \neg A''$, which is equivalent to the condition $A'' \cup G'' = \mathcal{B}$.

Due to the equivalence relation of contracts, we can provide a definition of assume-guarantee contracts over a universe of behaviors $\mathcal{B}$ by choosing one representative for each equivalence class, namely, the saturated contract. Our contracts over $\mathcal{B}$ are thus elements of the following set:

$$\{(A, G) \mid (A, G \subseteq \mathcal{B}) \wedge (A \cup G = \mathcal{B})\}. \tag{2}$$

2.2. Operations. Contracts are useful not only because they allow us to express assume-guarantee specifications of components and systems, but because they allow us to manipulate them. The algebra of contracts has binary operations with an analog in system design. For example, the binary operation of composition takes the contracts of components that are being interconnected and yields the contract for the system obtained from such interconnection. The operation of quotient solves the following problem: suppose we have a contract that a system under implementation should satisfy, and we also have the contract for a design element that will be used in the system; the quotient gives the most relaxed contract for the component that needs to be added to the system so that it satisfies the desired top-level objective. Information about various contract operations and how they are related can be found in [Incer Romeo, 2022, Chapter 6].

2.3. Languages and contracts. The definition of contracts given by (2) requires the expression of assumptions and guarantees as sets of behaviors. As writing properties as sets is impractical in engineering applications, we seek a description of contracts in terms of languages. We follow [Incer Romeo, 2022, Chapter 7] to do this.

As before, suppose $\mathcal{B}$ is the universe of behaviors modeled in the system. Let L be a language used to represent properties in the system. Many such languages are typical in engineering. Some examples are Linear Temporal Logic [Pnueli, 1977] and Signal Temporal Logic [Maler and Nickovic, 2004]. We require L to contain the logical operators $\wedge$, $\vee$, and $\neg$. We assume the existence of a denotation map $\mathrm{Den}\colon L \to 2^{\mathcal{B}}$ commuting with the logical operators:

$$\begin{aligned}\mathrm{Den}(\phi \wedge \psi) &= \mathrm{Den}(\phi) \cap \mathrm{Den}(\psi),\\ \mathrm{Den}(\phi \vee \psi) &= \mathrm{Den}(\phi) \cup \mathrm{Den}(\psi), \text{ and}\\ \mathrm{Den}(\neg\phi) &= \neg\mathrm{Den}(\phi) = \mathcal{B} \setminus \mathrm{Den}(\phi).\end{aligned}$$

If we understand two elements of L to be equal if they have the same denotation (as in Tarski-Lindenbaum algebras), L is a Boolean algebra. This motivates the following definition of assume-guarantee contracts over an arbitrary Boolean algebra:

2.4. Definition. *Let $B \in \mathbf{Obj}(\mathbf{Bool})$. A contract over B is an element of the set*

$$\mathbf{C}(B) = \{(a, g) \in B^{op} \times B \mid a \vee g = 1\}. \tag{3}$$

Even though contracts are generally understood as requirements having a specific form (i.e., a requirement coming in assume-guarantee form), this definition suggests that contracts can be more expressive than traditional requirements. An engineering requirement is a predicate over behaviors. Either the behavior satisfies or does not satisfy the requirement, such as being performant or safe. Let $\mathbf{2}$ be the 2-element lattice $\{\texttt{SAT}, \texttt{UNSAT}\}$ with $\texttt{UNSAT} \leq \texttt{SAT}$. For $l \in L$, we define the requirement

$$\mathcal{B} \to \mathbf{2} \quad b \mapsto \begin{cases}\texttt{SAT}, & \text{if } b \in \mathrm{Den}(l)\\ \texttt{UNSAT}, & \text{otherwise.}\end{cases}$$

Here the value `SAT` means the requirement expressed by l is satisfied; `UNSAT` means that it is not. In contrast, a contract $(a, g) \in \mathbf{C}(L)$ maps behaviors to the lattice $\mathbf{3} \coloneqq \{\texttt{FAIL}, \texttt{ACTIVE}, \texttt{IDLE}\}$, where $\texttt{FAIL} \leq \texttt{ACTIVE} \leq \texttt{IDLE}$, as follows:

$$\mathcal{B} \to \mathbf{3} \quad b \mapsto \begin{cases} \texttt{IDLE}, & \text{if } b \in \mathrm{Den}(\neg a \wedge g) \\ \texttt{ACTIVE}, & \text{if } b \in \mathrm{Den}(a \wedge g) \\ \texttt{FAIL}, & \text{if } b \in \mathrm{Den}(a \wedge \neg g). \end{cases}$$

The value `FAIL` means that the behavior satisfies the assumptions of the contract, but not the guarantees. We deem this a violation of the contract. `ACTIVE` means that the behavior satisfies both the assumptions and the guarantees of the contract. This is what we expect during the normal operation of a component. `IDLE` means that the behavior does not satisfy the assumptions of the contract. By (3), when the assumptions are not satisfied, the guarantees are satisfied automatically, thus matching the intuition behind assume-guarantee reasoning.

3. Contracts and Stone algebras

The purpose of this paper is to provide a universal characterization of contracts. This section introduces augmented Stone algebras as an abstraction of contracts. We will show eventually that contracts appear in an adjunction involving augmented Stone algebras and Boolean algebras. First, we recall some definitions.

3.1. Definition. [Borceux, 1994, Definition 1.2.1] *A Heyting algebra* $(H, 0, 1, \wedge, \vee, \to)$ *is a bounded lattice* $(H, 0_H, 1_H, \wedge, \vee)$ *such that, for every* $x \in H$*, the functor*

$$(\text{—}) \wedge x \colon H \to H \quad y \mapsto y \wedge x$$

has the right adjoint

$$x \to (\text{—}) \colon H \to H \quad z \mapsto (x \to z).$$

3.2. Remark. In a Heyting algebra H, we define the pseudo-complement of an element $x \in H$ as

$$\neg x \stackrel{\text{def}}{=} (x \to 0).$$

We define the *closure* of x as $\mathrm{Clos}(x) \stackrel{\text{def}}{=} \neg\neg x$ and say that x is *closed*[3] if $x = \mathrm{Clos}(x)$. We say that x is *dense* if $\mathrm{Clos}(x) = 1$.

3.3. Definition. [Borceux, 1994, Proposition 1.2.11] *A Boolean algebra is a Heyting algebra* $(B, \wedge, \vee, 1_B, 0_B, \to)$ *satisfying the identity* $\neg x \vee x = 1_B$ *for all* $x \in B$*.*

[3]Closed elements are also called *regular* in the literature [Borceux, 1994, Definition 1.2.12].

3.4. Definition. [Birkhoff, 1967, p. 130] *A Stone algebra is a Heyting algebra* $(S, \wedge, \vee, 1_S, 0_S, \to)$ *satisfying the identity* $\neg x \vee \mathrm{Clos}(x) = 1_S$ *for all* $x \in S$.

The identity $\neg x \vee x = 1$ is usually called *the law of excluded middle*. The identity $\neg x \vee \mathrm{Clos}(x) = 1$ is known as *weak excluded middle* and *De Morgan's law*[4]. De Morgan's law has important applications in topology through the notion of extremally disconnected spaces, defined as spaces whose lattice of open sets satisfies De Morgan's law [Johnstone, 1982, p. 102]. For example, [Gleason, 1958] shows that projective spaces exactly coincide with extremally disconnected spaces in the category of compact Hausdorff spaces with continuous maps.

These are some useful facts about Heyting algebras:

3.5. Proposition. *Let* H *be a Heyting algebra. For all* $x, y \in H$*, the following hold:*

(i) $x \leq \mathrm{Clos}(x)$*;*

(ii) $\mathrm{Clos}(\mathrm{Clos}(x)) = \mathrm{Clos}(x)$*;*

(iii) $\mathrm{Clos}(x \wedge y) = \mathrm{Clos}(x) \wedge \mathrm{Clos}(y)$*;*

(iv) $\mathrm{Clos}(x \to y) = \mathrm{Clos}(x) \to \mathrm{Clos}(y)$*;*

(v) x *admits a factorization* $x = c \wedge d$*, where* $c \in H$ *is closed and* $d \in H$ *is dense. Moreover,* $c = \mathrm{Clos}(x)$.

Proof. (i)-(iv) are shown in [Borceux, 1994, Prop. 1.2.8]. To prove (v), set $x_c = \mathrm{Clos}(x)$ and $x_d = (x_c \to x)$. Then

$$x_c \wedge x_d = x_c \wedge (x_c \to x) = x_c \wedge x \overset{\text{(i)}}{=} x.$$

x_c is closed due to (ii). We also have

$$\mathrm{Clos}(x_d) \overset{\text{(iv)}}{=} \mathrm{Clos}(x_c) \to \mathrm{Clos}(x) \overset{\text{(ii)}}{=} \mathrm{Clos}(x) \to \mathrm{Clos}(x) = 1,$$

so x_d is dense.

To prove the second part, suppose $x = c \wedge d$, where c is closed and d dense. We have $\mathrm{Clos}(x) = \mathrm{Clos}(c \wedge d) \overset{\text{(iii)}}{=} \mathrm{Clos}(c) \wedge \mathrm{Clos}(d) = \mathrm{Clos}(c) = c$. ∎

[4]De Morgan's law is shown to have several equivalent formulations in [Johnstone, 1979a] and in [Johnstone, 1979b].

3.6. Proposition. *Let B be a Boolean algebra. The contract algebra $\mathbf{C}(B)$ is a Stone algebra but not necessarily a Boolean algebra.*

Proof. First we will show that $\mathbf{C}(B)$ is a Heyting algebra. For $(a, g), (a', g') \in \mathbf{C}(B)$, Definition 3 yields the order $(a, g) \leq (a', g')$ iff $a' \leq a$ and $g \leq g'$. From this order, the meet and join given by (1) follow immediately. We verify the absorption laws of lattices:

$$(a, g) \wedge ((a, g) \vee (a', g')) = (a, g) \wedge (a \wedge a', g \vee g') = (a, g)$$
$$(a, g) \vee ((a, g) \wedge (a', g')) = (a, g) \vee (a \vee a', g \wedge g') = (a, g).$$

As discussed in Section 1, $0 = (1_B, 0_B)$ and $1 = (0_B, 1_B)$ are, respectively, the smallest and biggest elements of $\mathbf{C}(B)$. Thus, $\mathbf{C}(B)$ is a bounded lattice. That the functor $(—) \wedge (a, g) \colon \mathbf{C}(B) \to \mathbf{C}(B)$ is the left adjoint of the functor $(a, g) \to (—) \colon \mathbf{C}(B) \to \mathbf{C}(B)$ is shown in [Incer Romeo, 2022, Proposition 6.8.3]. Thus, $\mathbf{C}(B)$ is a Heyting algebra.

We now verify whether the contract algebra of B is a Boolean algebra or Stone algebra.

The pseudocomplement of $\mathbf{C}(B)$ is given by

$$\neg(a, g) \stackrel{\text{def}}{=} (a, g) \to 0 = (g, \neg g).$$

We verify the law of excluded middle:

$$(a, g) \vee \neg(a, g) = (a, g) \vee (g, \neg g) = (a \wedge g, 1_B).$$

Thus, $\mathbf{C}(B)$ is a Boolean algebra only when $a \wedge g$ are always equal to 0_B. This only happens when $0_B = 1_B$. However, $\mathbf{C}(B)$ satisfies De Morgan's law:

$$\neg(a, g) \vee \neg\neg(a, g) = (g, \neg g) \vee (\neg g, g) = (0_B, 1_B).$$

Thus, $\mathbf{C}(B)$ is a Stone algebra. ∎

As discussed in Section 2, when the elements a, g of a contract (a, g) evaluate to 0_B or 1_B, the contract (a, g) evaluates to either 0, $(1_B, 1_B)$, or 1—which we called `Fail`, `Active`, and `Idle`, respectively. 0 and 1 are the bounds of the lattice. What is $(1_B, 1_B)$?

3.7. Remark. The contract $(1_B, 1_B)$ is the smallest dense element of $\mathbf{C}(B)$. Observe that $\mathrm{Clos}(a, g) = (\neg g, g)$. Thus, for the contract (a, g) to be dense, we must have $g = 1_B$. This means that the largest dense contract in $\mathbf{C}(B)$ is $(0_B, 1_B) = 1$, and the smallest is $(1_B, 1_B)$. We will call this contract e.

3.8. Proposition. *In a Heyting algebra H, the following conditions are equivalent:*

(i) *H contains a smallest dense element e.*

(ii) *There exists an element $e \in H$ such that $e \to x = \mathrm{Clos}(x)$ for all $x \in H$.*

PROOF. (i) ⇒ (ii). Suppose that H has a minimum dense element e. Let $x \in H$. By Proposition 3.5.v, we can write x as $x = x_c \wedge x_d$ with $x_c = \mathrm{Clos}(x)$ closed and x_d dense. We have

$$\begin{aligned} e \to x &= e \to (x_c \wedge x_d) = (e \to x_c) \wedge (e \to x_d) \\ &= (e \to x_c) \overset{\text{Prop. 3.5.i}}{\leq} \mathrm{Clos}(e \to x_c) \overset{\text{Prop. 3.5.iv}}{=} \mathrm{Clos}(e) \to \mathrm{Clos}(x_c) = \mathrm{Clos}(x_c) = x_c \\ &\leq e \to x_c, \end{aligned}$$

where the third equality follows from the assumption that e is the minimum dense element. We have shown that $e \to x = \mathrm{Clos}(x)$.

(ii) ⇒ (i). Let e satisfy (ii) and $d \in H$ be dense. Then $1 = \mathrm{Clos}(d) = e \to d$, which means that $e \leq d$. Therefore, e is the minimum dense element. ∎

We use this result to define a class of Stone algebras.

3.9. DEFINITION. *A Stone algebra satisfying either condition of Proposition 3.8 will be called an* augmented Stone algebra. *We will refer to its element e as the* closure *element.*

3.10. NOTATION. Let **Bool** be the category of Boolean algebras, and **augStone** that of augmented Stone algebras. Morphisms of augmented Stone algebras are Heyting algebra morphisms that map the closure element in the domain to the closure element in the codomain. We define $\mathrm{Clos}\colon \mathbf{augStone} \to \mathbf{Bool}$ as the functor that maps an augmented Stone algebra to its Boolean algebra of closed elements, i.e., for $S \in \mathbf{Obj}(\mathbf{augStone})$, $\mathrm{Clos}(S) = \{\neg\neg x \mid x \in S\}$.

3.11. EXAMPLE. Any Boolean algebra B is an augmented Stone algebra with $e_B = 1_B$. The contract algebra $\mathbf{C}(B)$ is an augmented Stone algebra with $e = (1_B, 1_B)$. The locale of open sets of Sierpiński space is an augmented Stone algebra.[5]

We will use the following fact of Stone algebras.

3.12. PROPOSITION. *Let S be a Stone algebra. If $x \in S$ is closed, $x \to y = \neg x \vee y$.*

PROOF. For $a \in S$, we have $a \leq x \to y \Leftrightarrow a \wedge x \leq y \Leftrightarrow (a \wedge x) \vee \neg x \leq y \vee \neg x \Leftrightarrow (a \wedge x) \vee \neg x \vee (\neg x \wedge a) \leq y \vee \neg x \Leftrightarrow a \leq y \vee \neg x$. ∎

We are ready to characterize the functor $\mathbf{C}$.

3.13. PROPOSITION. $\mathbf{C}$ *is a functor* $\mathbf{C}\colon \mathbf{Bool} \to \mathbf{augStone}$.

[5]Sierpiński space is rich in topological properties. We leave as an open question whether some of these topological properties can be explained by the existence of a minimum dense element.

PROOF. Given $B \in \mathbf{Obj}(\mathbf{Bool})$, we know that $\mathbf{C}(B) \in \mathbf{Obj}(\mathbf{augStone})$ from Proposition 3.6 and Remark 3.7. Given a map $f \in \hom_{\mathbf{Bool}}(B, B')$, from the paragraph after Definition 1.1, we have $\mathbf{C}(f)(a, g) = (fa, fg) \in \mathbf{C}(B')$, as $f(a) \vee f(g) = f(a \vee g) = 1_{B'}$. Moreover, $\mathbf{C}(f)(1) = 1$, $\mathbf{C}(f)(0) = 0$, and $\mathbf{C}(f)(e) = e$. We also have

$$\begin{aligned}\mathbf{C}(f)((a, g) \wedge (a', g')) &= \mathbf{C}(f)(a \vee a', g \wedge g') = (f(a) \vee f(a'), f(g) \wedge f(g')) \\ &= \mathbf{C}(f)(a, g) \wedge \mathbf{C}(f)(a', g'), \\ \mathbf{C}(f)((a, g) \vee (a', g')) &= \mathbf{C}(f)(a \wedge a', g \vee g') = (f(a) \wedge f(a'), f(g) \vee f(g')) \\ &= \mathbf{C}(f)(a, g) \vee \mathbf{C}(f)(a', g'),\end{aligned}$$

and

$$\begin{aligned}\mathbf{C}(f)((a', g') \to (a, g)) &= \mathbf{C}(f)((g' \to g) \to (a \wedge \neg a'), g' \to g) \\ &= ((fg' \to fg) \to (fa \wedge \neg fa'), fg' \to fg) \\ &= \mathbf{C}(f)(a', g') \to \mathbf{C}(f)(a, g).\end{aligned}$$

This shows that $\mathbf{C}f \in \hom_{\mathbf{augStone}}(\mathbf{C}B, \mathbf{C}B')$. ∎

4. An adjunction between **Bool** and **augStone**

Now we show that $\mathbf{C}$ is the left adjoint of the functor $\mathrm{Clos}\colon \mathbf{augStone} \to \mathbf{Bool}$. Let $B \in \mathbf{Obj}(\mathbf{Bool})$ and $S \in \mathbf{Obj}(\mathbf{augStone})$. For $(a, g) \in \mathbf{C}(B)$, we let $\pi_1(a, g) = a$ and $\pi_2(a, g) = g$, and for $x \in B$, we let $\Delta x = (\neg x, x)$.

4.1. PROPOSITION. *Let $f \in \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S))$. The assignment*

$$\alpha_{B,S}\colon f \mapsto f\pi_2 \wedge (f\pi_1 \to e_S)$$

is a set morphism $\alpha_{B,S}\colon \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S)) \to \hom_{\mathbf{augStone}}(\mathbf{C}(B), S)$.

PROOF. Let $f^* = \alpha_{B,S} f$ and $c, c' \in \mathbf{C}(B)$, where $c = (a, g)$ and $c' = (a', g')$.

- $f^*(0) = f(0_B) \wedge (f(1_B) \to e_S) = 0_S$.
- $f^*(1) = f(1_B) \wedge (f(0_B) \to e_S) = 1_S$.
- $f^*(e) = f(1_B) \wedge (f(1_B) \to e_S) = e_S$.
- $f^*(c \wedge c') = f^*(a \vee a', g \wedge g') = f(g \wedge g') \wedge (f(a \vee a') \to e)$
  $= f(g) \wedge f(g') \wedge (f(a) \to e) \wedge (f(a') \to e) = f^*c \wedge f^*c'$.

- $f^*(c \vee c') = f^*(a \wedge a', g \vee g') = f(g \vee g') \wedge (f(a \wedge a') \to e)$
  $= (f(g) \wedge (f(a) \wedge f(a') \to e)) \vee (f(g') \wedge (f(a) \wedge f(a') \to e))$.
  Since $a' \vee g' = 1_B$, we have $\neg a' \leq g'$. Thus,

$$\begin{aligned} f^*(c \vee c') = &(f(g) \wedge (f(a) \wedge f(a') \to e)) \vee \\ &(f(g') \wedge (f(a) \wedge f(a') \to e)) \vee (\neg f(a') \wedge (f(a) \wedge f(a') \to e)) \\ = &(f(g) \wedge (f(a) \wedge f(a') \to e)) \vee (f(g') \wedge (f(a) \wedge f(a') \to e)) \\ &\vee \neg f(a'). \end{aligned}$$

  The last equality follows from the fact that $\neg f(a') \leq f(a) \wedge f(a') \to e$. Since $f(a')$ is closed, we apply Proposition 3.12 to conclude that

$$f^*(c \vee c') = (f(g) \wedge (f(a) \to e)) \vee (f(g') \wedge (f(a) \wedge f(a') \to e)). \tag{4}$$

  By applying an analogous procedure, we obtain

$$f^*(c \vee c') = (f(g) \wedge (f(a) \wedge f(a') \to e)) \vee (f(g') \wedge (f(a') \to e)).$$

  Conjoining this expression with (4) yields

$$\begin{aligned} f^*(c \vee c') &= (f(g) \wedge (f(a) \to e)) \vee (f(g') \wedge (f(a') \to e)) \\ &= f^*c \vee f^*c'. \end{aligned}$$

- $f^*(c' \to c)$
  $= f(g' \to g) \wedge (f((a \wedge \neg a') \vee \neg(g' \to g)) \to e)$
  $= f(g' \to g) \wedge (\neg f(a \wedge \neg a') \wedge f(g' \to g) \vee e)$ (by Proposition 3.12)
  $= f(g' \to g) \wedge (f(a \wedge \neg a') \to e)$
  $= f(g' \to g) \wedge f(\neg a' \to g) \wedge (f(a \wedge \neg a') \to e)$
  The last equality follows from the fact that $\neg a' \leq g'$. We have

$$\begin{aligned} &f^*(c' \to c) \\ &\quad = (f(g') \wedge e \to f(g) \wedge (f(a) \to e)) \wedge (\neg f(a') \to f(g) \wedge (f(a) \to e)), \end{aligned}$$

  which holds because $f(g' \to g) = (f(g') \wedge e \to f(g) \wedge (f(a) \to e))$. We finally obtain

$$\begin{aligned} f^*(c' \to c) &= (\neg f(a') \vee f(g') \wedge e) \to (f(g) \wedge (f(a) \to e)) \\ &= f(g') \wedge (f(a') \to e) \to (f(g) \wedge (f(a) \to e)) \\ &= f^*c' \to f^*c. \end{aligned}$$

Since f^* commutes with the binary operations and distinguished elements, the conclusion of the proposition holds. ∎

4.2. PROPOSITION. *Let* $f^* \in \hom_{\mathbf{augStone}}(\mathbf{C}(B), S)$. *The assignment*

$$\beta_{B,S}\colon f^* \mapsto f^*\Delta$$

is a set morphism $\beta_{B,S}\colon \hom_{\mathbf{augStone}}(\mathbf{C}(B), S) \to \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S))$. *Moreover,* $\alpha_{B,S}$ *and* $\beta_{B,S}$ *are inverses.*

PROOF. Let $b, b' \in B$ and set $f = \beta_{B,S}f^*$. We verify that f is a morphism in **Bool**:

- $f(0_B) = f^*\Delta 0_B = f^*0 = 0_S$.
- $f(1_B) = f^*\Delta 1_B = f^*1 = 1_S$.
- $f(b \wedge b') = f^*(\neg(b \wedge b'), b \wedge b') = (f^*\Delta b) \wedge (f^*\Delta b') = fb \wedge fb'$.
- $f(b \vee b') = f^*(\neg(b \vee b'), b \vee b') = (f^*\Delta b) \vee (f^*\Delta b') = fb \vee fb'$.
- $f(b' \to b) = f^*(\neg(b' \to b), b' \to b) = f^*\left((\neg b', b') \to (\neg b, b)\right) = fb' \to fb$.

Regarding the second part, let $(a, g) \in \mathbf{C}(B)$. We have

$$\begin{aligned}(\alpha_{B,S}\beta_{B,S}f^*)(a,g) &= (\alpha_{B,S}f^*\Delta)(a,g) = (f^*\Delta\pi_2 \wedge (f^*\Delta\pi_1 \to e_S))(a,g)\\ &= f^*(\neg g, g) \wedge (f^*(\neg a, a) \to e_S) = f^*\left((\neg g, g) \wedge ((\neg a, a) \to e)\right)\\ &= f^*\left((\neg g, g) \wedge (a, 1)\right) = f^*(a, g)\end{aligned}$$

and

$$\begin{aligned}(\beta_{B,S}\alpha_{B,S}f)(b) &= (f\pi_2\Delta \wedge (f\pi_1\Delta \to e_S))(b) = f(b) \wedge (f(\neg b) \to e_S)\\ &= f(b) \wedge (f(b) \vee e_S) = f(b).\end{aligned}$$

■

4.3. COROLLARY. *The functor* $\mathbf{C}$ *is fully faithful.*

PROOF. Let B and B' be Boolean algebras. The maps $\Delta\colon B \to \mathrm{Clos}(\mathbf{C}(B))$ and $\pi_2\colon \mathrm{Clos}(\mathbf{C}(B)) \to B$ are inverses in **Bool**. Therefore,

$$\hom_{\mathbf{Bool}}(B, B') \cong \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(\mathbf{C}(B'))) \cong \hom_{\mathbf{augStone}}(\mathbf{C}(B), \mathbf{C}(B')),$$

where the second isomorphism follows from Proposition 4.2. ■

4.4. Remark. While $\mathbf{C}$ is fully faithful, $\mathbf{C}$ does not establish a surjection between the objects of **Bool** and **augStone**. Let B_0 be the Boolean domain (i.e., a Boolean algebra with only two elements, 0 and 1). The cardinality of its contract algebra is $|\mathbf{C}(B)| = 3$. The smallest Boolean algebra B_1 of cardinality larger than 2 has four elements: $0, 1, a, \neg a$ for some atom $a \in B_1$. Its contract algebra has cardinality $|\mathbf{C}(B_1)| = 9$. However, the algebra $\{0, e, x, 1\}$ with $e \leq x$ is an augmented Stone algebra and has 4 elements. Our argument shows that there is no Boolean algebra that $\mathbf{C}$ maps to an algebra of this cardinality.

4.5. Theorem. *The functor* $\mathbf{C}$ *is the left adjoint of* Clos.

Proof. Let $B \in \mathbf{Obj}(\mathbf{Bool})$ and $S \in \mathbf{Obj}(\mathbf{augStone})$. By Propositions 4.1 and 4.2, $\alpha_{B,S} \colon \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S)) \to \hom_{\mathbf{augStone}}(\mathbf{C}(B), S)$ and $\beta_{B,S} \colon \hom_{\mathbf{augStone}}(\mathbf{C}(B), S) \to \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S))$ are inverses. It remains to show these isomorphisms are natural in both arguments. Let $\rho \in \hom_{\mathbf{Bool}}(B', B)$ and $\sigma \in \hom_{\mathbf{augStone}}(S, S')$. We want to show that the following diagram commutes:

$$\begin{array}{ccc} \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S)) & \underset{\beta_{B,S}}{\overset{\alpha_{B,S}}{\rightleftarrows}} & \hom_{\mathbf{augStone}}(\mathbf{C}(B), S) \\ \downarrow{\scriptstyle \sigma \circ () \circ \rho} & & \downarrow{\scriptstyle \sigma \circ () \circ \mathbf{C}(\rho)} \\ \hom_{\mathbf{Bool}}(B', \mathrm{Clos}(S')) & \underset{\beta_{B',S'}}{\overset{\alpha_{B',S'}}{\rightleftarrows}} & \hom_{\mathbf{augStone}}(\mathbf{C}(B'), S') \end{array}$$

Let $f \in \hom_{\mathbf{Bool}}(B, \mathrm{Clos}(S))$ and $(a', g') \in \mathbf{C}(B')$. We have

$$\begin{aligned} \left(\alpha_{B',S'}(\sigma f \rho)\right)(a', g') &= (\sigma f \rho \pi_2 \wedge (\sigma f \rho \pi_1 \to e_{S'}))(a', g') \\ &= \sigma f \rho(g') \wedge (\sigma f \rho(a') \to e_{S'}) \\ &= \sigma \left(f \rho(g') \wedge (f \rho(a') \to e_S)\right) \\ &= \sigma \left(f \pi_2 \wedge (f \pi_1 \to e_S)\right)(\rho(a'), \rho(g')) \\ &= \sigma \alpha_{B,S}(f) \mathbf{C}(\rho)(a', g'). \end{aligned}$$

Let $f^* \in \hom_{\mathbf{augStone}}(\mathbf{C}(B), S)$ and $b' \in B'$. We have

$$\begin{aligned} \beta_{B',S'}\left(\sigma f^* \mathbf{C}(\rho)\right)(b') &= \sigma f^* \mathbf{C}(\rho) \Delta(b') \\ &= \sigma f^*(\neg \rho b', \rho b') \\ &= \sigma f^* \Delta(\rho b') \\ &= (\sigma \beta_{B,S}(f^*) \Delta \rho)(b'). \end{aligned}$$

■

5. Discussion and concluding remarks

Our objective in this paper was to provide a universal characterization of contracts. This was done in Theorem 4.5, showing that contracts appear as the left adjoint of a closure functor. As adjoint functors are unique, and the closure functor is canonical, this result shows that contracts are fundamental structures that come with Boolean algebras. Considering the increasing importance of contracts in engineering applications, this result validates the soundness of the definition of contracts that has been adopted. We point out that it has also been recently shown that contracts can be understood as bounded Sugihara monoids [Castiglioni and Ertola-Biraben, 2024]. There are two immediate directions for future work.

5.1. The contract topos? One of the ways in which toposes may play a role in engineering design is through their ability to carry syntax and semantics in the same structure. To what extent can our present discussion be ported to the domain of toposes? We saw that a given Boolean algebra has an associated contract algebra, which happens to be a Stone algebra. Can we rewrite this last sentence by replacing the word *algebra* with the word *topos*? Assuming that the contract topos exists for arbitrary Boolean toposes, how is this construction generalized when the toposes are not Boolean?

5.2. A complex of Stone algebras? It is well-known that the inclusion functor is the right adjoint of the closure, so we have the following diagram.

$$\mathbf{Bool} \underset{\mathbf{C}}{\overset{\iota}{\underset{\text{Clos}}{\overset{\top}{\leftarrow}}}} \mathbf{augStone} \qquad (5)$$

5.3. Remark. The definition of contracts (Definition 2.4) is akin to that of the standard simplex $\Delta^1 := \{(t_0, t_1) \in \mathbb{R}^2 \mid 0 \leq t_i \leq 1, t_0 + t_1 = 1\}$. Also, (5) is reminiscent of the beginning of a simplicial set, so we wonder how it could be extended to the right.

With an eye to address the latter point, it helps to consider the initial objects of the two categories involved in (5). The initial element of **Bool** is $\mathbf{2} := \{\bullet \to \bullet\}$. The initial element of **augStone** is $\mathbf{3} := \{\bullet \to \bullet \to \bullet\}$. This suggests that we seek other subcategories of Stone algebras having as initial element $\mathbf{n}$. For ease of definition, we will refer to the bottom of the Stone algebras as e_0.

- Let $\mathbf{S_0}$ be the subcategory of Stone algebras for which the following closure operator is the identity:
$$\text{Clos}_0 := \neg_0\neg_0, \text{ where } \neg_0 := (\text{—}) \to e_0.$$
Clearly, $\mathbf{S_0}$ is just another name for **Bool**, and Clos_0 for Clos.

- For $n \geq 1$, let $\mathbf{S_n}$ be the subcategory of Stone algebras having distinguished elements $e_1, \ldots, e_n$ with $e_1 \leq e_2 \leq \ldots \leq e_n$ for which the following closure operator is the

identity:

$$\mathrm{Clos}_n \coloneqq \bigwedge_{i=0}^{n} \neg_i\neg_i = \mathrm{Clos}_{n-1} \wedge \neg_n\neg_n, \text{ where } \neg_i \coloneqq (\text{—}) \to e_i. \tag{6}$$

This definition allows us to speak of degrees of closure. For an element $x \in S \in \mathbf{Obj}(\mathbf{S_n})$ (with $n \geq 0$), we can say that x is *i-closed* if $\mathrm{Clos}_i(x) = x$, and *i-dense* if $\mathrm{Clos}_i(x) = 1$. Nay, we can define the *dimension* of x to be the smallest i such that x is i-closed.

Let us now list some useful facts to reason about these operators:

5.4. Proposition. *Let H be a Heyting algebra and $a \in H$. Let $\neg_a \coloneqq (\text{—}) \to a$. For all $x, y \in H$, the following hold:*

(i) *if $x \leq y$, $\neg_a y \leq \neg_a x$;*

(ii) $x \leq \neg_a\neg_a x$;

(iii) $\neg_a = \neg_a\neg_a\neg_a$;

(iv) $\neg_a(x \vee y) = \neg_a x \wedge \neg_a y$;

(v) $\neg_a\neg_a(x \wedge y) = \neg_a\neg_a x \wedge \neg_a\neg_a y$.

Proof. The arguments are very similar to the proofs in [Borceux, 1994, Prop. 1.2.8]. They are as follows:

(i) Suppose $x \leq y$. Then $\neg_a y \wedge y = y \wedge a \leq a \Rightarrow \neg_a y \wedge x \leq a \Leftrightarrow \neg_a y \leq \neg_a x$.

(ii) $x \wedge \neg_a x = x \wedge a \leq a \Rightarrow x \leq \neg_a\neg_a x$.

(iii) From (ii), it follows that $\neg_a x \leq \neg_a\neg_a\neg_a x$. We also have $\neg_a\neg_a\neg_a x \wedge x \leq \neg_a\neg_a\neg_a x \wedge \neg_a\neg_a x \leq a \Rightarrow \neg_a\neg_a\neg_a x \leq \neg_a x$.

(iv) Let $z \in H$. Then

$$\begin{aligned} z \leq \neg_a(x \vee y) &\Leftrightarrow (z \wedge x) \vee (z \wedge y) \leq a \\ &\Leftrightarrow (z \wedge x) \leq a \text{ and } (z \wedge y) \leq a \\ &\Leftrightarrow z \leq \neg_a x \text{ and } z \leq \neg_a y \Leftrightarrow z \leq \neg_a x \wedge \neg_a y. \end{aligned}$$

(v) $x \wedge y \leq x$ and $x \wedge y \leq y$, so $\neg_a\neg_a(x \wedge y) \leq \neg_a\neg_a x$ and $\neg_a\neg_a(x \wedge y) \leq \neg_a\neg_a y$, which means that $\neg_a\neg_a(x \wedge y) \leq \neg_a\neg_a x \wedge \neg_a\neg_a y$.

We also have

$$\begin{aligned} \neg_a(x \wedge y) \wedge x \wedge y \leq a &\Leftrightarrow \neg_a(x \wedge y) \wedge x \leq \neg_a y = \neg_a\neg_a\neg_a y \\ &\Leftrightarrow \neg_a(x \wedge y) \wedge x \wedge \neg_a\neg_a y \leq a \\ &\Leftrightarrow \neg_a(x \wedge y) \wedge \neg_a\neg_a y \leq \neg_a x = \neg_a\neg_a\neg_a x \\ &\Leftrightarrow \neg_a(x \wedge y) \wedge \neg_a\neg_a x \wedge \neg_a\neg_a y \leq a \\ &\Leftrightarrow \neg_a\neg_a x \wedge \neg_a\neg_a y \leq \neg_a\neg_a(x \wedge y). \end{aligned}$$

■

5.5. PROPOSITION. *Let $S \in \mathbf{Obj}(\mathbf{S_n})$ (with $n > 0$) and $x, y \in S$. The following properties hold:*

(i) $e_i \to x = \mathrm{Clos}_{i-1}(x)$ *for all* $1 \leq i \leq n$;

(ii) $\mathrm{Clos}_i \circ \mathrm{Clos}_j(x) = \mathrm{Clos}_{\min\{i,j\}}(x)$ *for all* $0 \leq i, j \leq n$;

(iii) $\mathrm{Clos}_0(x) \geq \mathrm{Clos}_1(x) \geq \ldots \geq \mathrm{Clos}_n(x) = x$;

(iv) $\mathrm{Clos}_i(x \wedge y) = \mathrm{Clos}_i(x) \wedge \mathrm{Clos}_i(y)$ *for all* $0 \leq i \leq n$;

(v) $\mathrm{Clos}_i(x \to y) = \mathrm{Clos}_i(x) \to \mathrm{Clos}_i(y)$ *for all* $0 \leq i \leq n$;

(vi) *for $i \leq j$, x is j-closed if it is i-closed, and x is i-dense if it is j-dense;*

(vii) *for all $i \leq n$, x admits a factorization $x = c \wedge d$, where $c \in S$ is i-closed and $d \in S$ is i-dense. Moreover, $c = \mathrm{Clos}_i(x)$.*

PROOF.

(i) We observe that

$$e_i \to x = e_i \to \bigwedge_{k=0}^{n} ((x \to e_k) \to e_k) = \bigwedge_{k=0}^{n} ((x \to e_k) \to (e_i \to e_k))$$
$$= \bigwedge_{k=0}^{i-1} ((x \to e_k) \to (e_k)) = \mathrm{Clos}_{i-1}(x).$$

(ii) For $0 \leq i, j < n$, $\mathrm{Clos}_i \circ \mathrm{Clos}_j(x) = e_{i+1} \to (e_{j+1} \to x) = (e_{i+1} \wedge e_{j+1}) \to x = e_{1+\min\{i,j\}} \to x = \mathrm{Clos}_{\min\{i,j\}}(x)$. If $j = n$ and $0 \leq i \leq n$, $\mathrm{Clos}_i \circ \mathrm{Clos}_j(x) = \mathrm{Clos}_i(x) = \mathrm{Clos}_{\min\{i,n\}}(x)$. If $i = n$ and $0 \leq j < n$, $\mathrm{Clos}_i \circ \mathrm{Clos}_j(x) = (e_n \to (e_{j+1} \to x)) \wedge \neg_n \neg_n \mathrm{Clos}_j(x) = \mathrm{Clos}_j(x) \wedge \neg_n \neg_n \mathrm{Clos}_j(x) = \mathrm{Clos}_j(x)$.

(iii) The result follows from (6) and Proposition 5.4.ii.

(iv) Follows from (6) and Proposition 5.4.v.

(v) The condition holds trivially for $i = n$. For $i < n$, $\mathrm{Clos}_i(x \to y) = e_{i+1} \to (x \to y) = (e_{i+1} \wedge x) \to y = (e_{i+1} \wedge (e_{i+1} \to x)) \to y = (e_{i+1} \to x) \to (e_{i+1} \to y) = \mathrm{Clos}_i(x) \to \mathrm{Clos}_i(y)$.

(vi) Both statements follow from (iii).

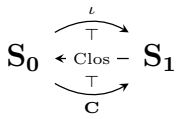
(vii) Let $x_c = \mathrm{Clos}_i(x)$ and $x_d = x_c \to x$. Then x_c is i-closed by ii, and x_d is i-dense by v. We have $x = x_c \wedge x_d$ by iii. Moreover, if $x = c \wedge d$, where c is i-closed and d is i-dense, $\mathrm{Clos}_i(x) = c$ by (iv).

■

This proposition shows that the closure operators have a great deal of structure. For $n \geq 1$, we have n functors $\mathrm{Clos}_i \colon \mathbf{S_n} \to \mathbf{S_{n-1}}$ $(0 \leq i < n)$. We also have the inclusion functor $\iota \colon \mathbf{S_{n-1}} \to \mathbf{S_n}$ and know that $\mathrm{Clos}_{n-1} \dashv \iota$. Finally, by Proposition 3.13, for $S \in \mathbf{Obj}(\mathbf{S_0})$, it is the case that $\mathbf{C}(S) \in \mathbf{augStone}$. Let $(a, g) \in \mathbf{C}(S)$. Then

$$\mathrm{Clos}_1(a, g) = \mathrm{Clos}_0(a, g) \wedge \neg_e \neg_e (a, g) = (\neg g, g) \wedge (a, 1_S) = (a, g).$$

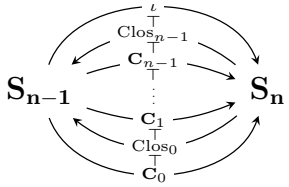
This means that $\mathbf{C}(S) \in \mathbf{Obj}(\mathbf{S_1})$. We can thus write (5) as



(7)

How to generalize (7) for any $n \geq 1$ is an open question. More specifically, we would like to answer the following:

- Are there functors $\mathbf{C}_i$ satisfying the diagram below?



(8)

- Can these functors be understood as higher-dimensional contracts, i.e., as the analog of higher-dimensional simplices—see Remark 5.3?
- Can these notions lead to a (co)chain complex and to (co)homology of Stone algebras?

References

[Benveniste et al., 2008] Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., and Sofronis, C. (2008). Multiple viewpoint contract-based specification and design. In de Boer, F. S., Bonsangue, M. M., Graf, S., and de Roever, W.-P., editors, *Formal Methods for Components and Objects: 6th International Symposium, FMCO 2007, Amsterdam, The Netherlands, October 24-26, 2007, Revised Lectures*, pages 200–225, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Benveniste et al., 2018] Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., Raclet, J.-B., Reinkemeier, P., Sangiovanni-Vincentelli, A., Damm, W., Henzinger, T. A., and Larsen, K. G. (2018). Contracts for system design. *Foundations and Trends® in Electronic Design Automation*, 12(2-3):124–400.

[Birkhoff, 1967] Birkhoff, G. (1967). *Lattice theory*, volume Vol. XXV of *American Mathematical Society Colloquium Publications*. American Mathematical Society, Providence, RI, third edition.

[Borceux, 1994] Borceux, F. (1994). *Handbook of categorical algebra. 3*, volume 52 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge. Categories of sheaves.

[Castiglioni and Ertola-Biraben, 2024] Castiglioni, J. L. and Ertola-Biraben, R. (2024). Assume-guarantee contract algebras are bounded Sugihara monoids. *arXiv 2402.12514*.

[Gleason, 1958] Gleason, A. M. (1958). Projective topological spaces. *Illinois J. Math.*, 2:482–489.

[Incer, 2023] Incer, I. (2023). An adjunction between Boolean algebras and a subcategory of Stone algebras. *arXiv 2309.04135*.

[Incer Romeo, 2022] Incer Romeo, I. (2022). *The Algebra of Contracts*. PhD thesis, EECS Department, University of California, Berkeley.

[Johnstone, 1979a] Johnstone, P. T. (1979a). Conditions related to De Morgan's law. In *Applications of sheaves (Proc. Res. Sympos. Appl. Sheaf Theory to Logic, Algebra and Anal., Univ. Durham, Durham, 1977)*, volume 753 of *Lecture Notes in Math.*, pages 479–491. Springer, Berlin.

[Johnstone, 1979b] Johnstone, P. T. (1979b). Another condition equivalent to De Morgan's law. *Comm. Algebra*, 7(12):1309–1312.

[Johnstone, 1982] Johnstone, P. T. (1982). *Stone spaces*, volume 3 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge.

[Maler and Nickovic, 2004] Maler, O. and Nickovic, D. (2004). Monitoring temporal properties of continuous signals. In Lakhnech, Y. and Yovine, S., editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Pnueli, 1977] Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)(FOCS)*, pages 46–57.

Electrical and Computer Engineering
University of Michigan
Ann Arbor, MI 48109
Email: `iir@umich.edu`

This article may be accessed at `http://www.tac.mta.ca/tac/`

THEORY AND APPLICATIONS OF CATEGORIES will disseminate articles that significantly advance the study of categorical algebra or methods, or that make significant new contributions to mathematical science using categorical methods. The scope of the journal includes: all areas of pure category theory, including higher dimensional categories; applications of category theory to algebra, geometry and topology and other areas of mathematics; applications of category theory to computer science, physics and other mathematical sciences; contributions to scientific knowledge that make use of categorical methods.
Articles appearing in the journal have been carefully and critically refereed under the responsibility of members of the Editorial Board. Only papers judged to be both significant and excellent are accepted for publication.

SUBSCRIPTION INFORMATION Individual subscribers receive abstracts of articles by e-mail as they are published. To subscribe, send e-mail to `tac@mta.ca` including a full name and postal address. Full text of the journal is freely available at `http://www.tac.mta.ca/tac/`.

INFORMATION FOR AUTHORS LaTeX2e is required. Articles may be submitted in PDF by email directly to a Transmitting Editor following the author instructions at `http://www.tac.mta.ca/tac/authinfo.html`.