Idempotent Resources in Separation Logic
The Heart of `core` in Iris

Daniel Gratzer, Mathias Adam Møller, and Lars Birkedal

Aarhus University, Aarhus, Denmark
gratzer@cs.au.dk 201704440@post.au.dk birkedal@cs.au.dk

Abstract. We revisit the foundational notion of "resources" used by separation logics from a categorical and algebraic viewpoint. In particular, we show that the *cameras* used by concurrent, higher-order, impredicative separation logics like Iris as a generalization of partial commutative monoids can be simplified and clarified and we introduce a category of cameras in which many vital cameras exhibit simple universal properties. We do this by observing that an important structure on cameras (the *core* operator) can be uniquely constrained and replaced by the property governing the idempotent elements of the camera. We verify that all cameras used in practice in Iris satisfy this property and use this insight to simplify the existing Iris formalization.

1 Introduction

Since its introduction in the late 1990s and early 2000s separation logics [20, 22], have gone through numerous conceptual revisions and alterations. Modern higher-order concurrent separation logics such as Iris [16] are now vastly more complex and built on top of several layers of abstraction in order to account for concurrency and higher-order reasoning, but the premise is still fundamentally unchanged: specifications for a program are given by pre- and post-conditions which are some form of predicate on program state and a "disjoint union" operator on program state gives rise to a separating conjunction in these specifications.

To manage this complexity, Iris attempts to realize the vast majority of the program logic purely within a flexible base logic. The result is that base Iris is a standard higher-order logic supplemented with the connectives of bunched implication logic, a handful of modalities, and a family of propositions denoting ownership. The heart of Iris is therefore in its highly abstracted definition of resource used by ownership propositions. These must be flexible enough to handle not only program state but also to encode the various features (invariants, state transition systems, etc.) that program logics utilize to verify programs.

Our goal is to revisit these abstract resources, termed *cameras*[1] in Iris, and isolate a new property of these structures. We capitalize on this new property to remove some of complexity of cameras and obtain a simpler notion of *resource algebras*. We then construct a category of resource algebras and show that important cameras in Iris admit simple universal properties as resource algebras.

[1] The name derives from an older acronym CMRA which proved inaccurate for Iris 3.0.

1.1 From heaps to cameras

The impetus for the more general notion of resource in Iris traces back to the advent of separation logic [22]. In particular, separation logic's eponymous separating conjunction is derived from an important piece of additional structure carried by program heaps (the most basic realization of program state): a disjoint union operation. Recall that one may represent heaps as partial maps from locations (e.g., natural numbers) to values. Given two such heaps h_1, h_2 with disjoint domains of definition, one may unambiguously form their union $h_1 \uplus h_2$. With this to hand, one defines the separating conjunction as follows:

$$(P * Q)\, h \triangleq \exists h_1, h_2.\, P\, h_1 \wedge Q\, h_2 \wedge h = h_1 \uplus h_2$$

While predicates in the original separation logic were taken over exactly the data needed to execute a program—i.e., the state of the heap—it was quickly realized that this was inadequate for realistic programs. Their correctness often depended upon invariants or conditions that might not be reflected directly in the memory. While a variety of solutions were proposed, the most elegant was to simply enlarge the definition of "program state" to include *ghost state*: objects which were not strictly necessary to describe the program execution. A simple example of ghost state is heaps with fractional permissions, which can be used to account for whether a thread has read or write ownership of a heap location [5].

The unifying feature of this more general ghost state is a partial 'union' operation. Whatever ghost state is used, it forms a set M equipped with a partial commutative and associative operation $\cdot$ (see, e.g., [6, 8, 9, 14]). Our definition of $*$ generalizes to such an M without change[2] whereby we can define the primitives of separation logic over an arbitrary *partial commutative monoid* M. This extra generality allows for a more conceptual definition of ghost state.

This story is complicated by higher-order state, concurrency, and other features. To cope with them, separation logics leverage *step-indexing* [1]. Rather than pre- and post-conditions being drawn from (monotone) predicates $M \to \mathrm{Prop}$, they are further indexed by a natural number $\mathbb{N} \times M \to \mathrm{Prop}$. The extra indexing soundly simulates arbitrary recursion in predicates.

In fact, embracing a small amount of revisionism, one can see all of this as ordinary separation logic done *internally* to a particular presheaf topos $\mathbf{PSh}(\omega)$ [3, 10, 19]. Working internally like this, it becomes natural to regard the partial commutative monoid as a "step-indexed" set with partial operations i.e., a partial commutative monoid internally to $\mathbf{PSh}(\omega)$. With this extra layer of generality, Jung et al. [16] showed that all the features of a modern concurrent separation logic could be built on top of ghost state within a higher-order logic.

With this motivation, we give a slightly imprecise definition of an Iris camera:

Definition 1. *A camera consists of a (step-indexed) set M, a (step-indexed) partial multiplication operation $(\cdot)$, and a partial operation $|-|$ which (when defined) maps m to an idempotent element $|m|$ such that $|m| \cdot m = m$.*

[2] In fact, it is now more recognizable as *Day convolution* [7]

1.2 Core and duplicable propositions

Unfortunately, the definition of camera is far more complex than the above might indicate. As a result, it is difficult to give a high-level definition of cameras without eliding important details and it is challenging to articulate why the camera axioms are in any way complete.[3] One manifestation of this poor behavior is the lack of a suitable category of cameras: there is no well-behaved notion of morphism of cameras and even simple constructions like the product of two cameras $M_1 \times M_2$ lack any universal property which could give a more conceptual description.

Some of the complexity is not, strictly speaking, mathematical. For instance, representing partial operations and step-indexing in a proof assistant is known to be arduous and the definition of camera is tuned to make it as easy as possible to realize in Coq at the cost of more uniform behavior. However, a central aspect of the generalization of cameras over partial commutative monoids is to replace the unit element ϵ with a *core operation* $|-|$ which is both complex and slightly ad-hoc. As it stands, requiring the existence of this operator is a major obstacle towards a good category of cameras.

To properly situate the definition of core and motivate our proposed replacement for it, we once more turn to classical separation logic. In this setting, there is a *comonadic modality* $\Box$ defined by:

$$(\Box P)\, h \triangleq P(\emptyset)$$

The $\Box$ modality occurs frequently when working within separation logic: it isolates those specifications for which separating conjunction coincides with ordinary conjunction such that, e.g., $\Box P$ can be duplicated to $\Box P * \Box P$. In Iris, for instance, $\Box$ is crucial for specifying features like *invariants* as well as for embedding ordinary propositions into the program logic. In the simple setting of classical separation logic, $\Box$ can actually be characterized in a number of distinct ways and any of which could be used to generalize $\Box$ to arbitrary cameras:

- $\Box$ is defined by evaluating at the *unit* of $\uplus$.
- $\Box$ is the "always" comonad from Kripke models of modal logic.
- $\Box$ is the unique operation preserving $\top$, $\wedge$, and $\exists$ such that $\Box(\ell \hookrightarrow v) = \bot$.
- $\Box$ is defined by evaluating at the unique *idempotent* with respect to $\uplus$.

In order to lift $\Box$ to an arbitrary camera M, the definition of a camera more-or-less hard-codes the necessary structure for the fourth definition; the axioms of $|-|$ are exactly those needed to ensure that $P \mapsto P \circ |-|$ (modulo partiality) gives rise to a $\Box$ modality.

Even this description hides some complexity. Bizjak and Birkedal [4] observed that absent step-indexing, there is a bijection between core structures and well-behaved $\Box$ modalities. However, this (1) does not extend to the step-indexed case and (2) offers no insight as to which of the many $\Box$ modalities should be chosen. In particular, when working with Heap there was a single clear choice of $\Box$ modality with a variety of different characterizations. In contrast, for a general camera there are many distinct choices of core inducing distinct $\Box$ modalities.

[3] Indeed, they differ between versions of Iris!

1.3 The maximal idempotent axiom

In this work, we simplify the definition of cameras by *removing* the core operation from their specification entirely. In its place, we instead impose a single additional property on our cameras insisting that for each $a \in M$, there exists a maximal idempotent below a or no idempotents at all. To disambiguate, we term the resulting structure a *resource algebra*. In other words, rather than requiring each resource algebra to come equipped with a partial assignment of elements to idempotents below them, we require that there is always a 'best' such assignment which picks out the largest possible idempotent. This choice in turn yields a $\Box$ modality uniquely defined as the largest modality satisfying the expected axioms.

This substantially simplifies the definition of a resource algebra: we exchange one operation and four axioms for this single new axiom. More than this, however, the removal of core makes it possible to give a satisfactory *category of resource algebra* **RA**. Given that cameras are nearly partial commutative monoids, it was long suspected that they should organize into a well-behaved category. In particular, it was conjectured that various examples of cameras used in Iris (sums, products, etc.) ought to enjoy universal properties within this category. We confirm this conjecture and observe that all of the cameras by Iris can be realized as resource algebras and that many common cameras induce resource algebras arising from adjoints or universal constructions within **RA**.

Our development of the theory of resource algebras takes place entirely within the internal language of $\mathbf{PSh}(\omega)$ and, consequently, does not mention step-indexing at all. In fact, our proofs are designed to apply also to transfinite versions of Iris (based on presheaves over larger ordinals) or separation logics without any step-indexing at all. The only cost for this generality is the need to work within a constructive setting.

1.4 Contributions

We contribute a simpler and more abstract definition of "resources" for concurrent separation logics like Iris. We do this by *removing* a structure from Iris's definition of camera and replacing it with a new axiom (the maximal idempotent axiom), thereby obtaining the new notion of a *resource algebra*. We show the following:

- Every camera used in Iris satisfies the maximal idempotent axiom and therefore induces a resource algebra.
- Every resource algebra induces a universal $\Box$ modality.
- Resource algebras assemble into a well-behaved category **RA** in which many important resource algebras enjoy simple universal properties.

All of our results are carried out within a version of extensional type theory and apply not just to step-indexing over ω but to transfinite separation logics as well.

Finally, we show how to adapt the Iris formalization in Coq as it stands today to partially benefit from these results. In so doing, we also note various places where expediency of formalization has complicated the definition of cameras and give a precise account of the trade-offs these induce.

2 A brief reprise of $\mathbf{PSh}(\omega)$

For much of this paper (Sections 3 to 5) we work *internally* to the topos of trees $\mathbf{PSh}(\omega)$. In this section, we review the basic properties of this internal language.

2.1 Type theory within $\mathbf{PSh}(\omega)$

To begin with, we recall that $\mathbf{PSh}(\omega)$ is a *presheaf topos* and therefore supports a rich *internal language*: a model of extensional dependent type theory with all the standard connectives, including a hierarchy of universes $(\mathcal{U}_i)$ [11, 12]. Among others, included in these are various inductive types such as the type of natural numbers, lists, and so on. We note that as we are dealing with extensional equality, both function extensionality (funext) and unicity of identity proofs (UIP) hold. As is standard, we largely ignore size issues and simply write $\mathcal{U}$. Also included is a type of propositions Prop closed under all the standard logical connectives, including impredicative quantification. There is a function $\mathrm{Prf} : \mathrm{Prop} \to \mathcal{U}$ which sends a proposition to the type of its proofs such that $\mathrm{Prf}(\phi)$ contains at most one element. Finally, Prop satisfies propositional extensionality: $\mathrm{Prf}(\phi) \cong \mathrm{Prf}(\psi) \iff \phi = \psi$.

Rather than explicitly adding all the necessary logical connectives to Prop, we realize them all at once through the addition of an "inverse" of sorts to Prf sending a type $A : \mathcal{U}_i$ to the propositional truncation $[A] : \mathrm{Prop}$. The defining characteristic of propositional truncation is that if B is a type with at most one element, then $A \to B \cong \mathrm{Prf}([A]) \to B$. Consequently, $\mathrm{Prf}([A]) \cong A$ if A is a type containing at most one element e.g., if $A = \mathrm{Prf}(\phi)$. One can then show, for instance, that logical entailment $\phi \to \psi$ can be defined as $[\mathrm{Prf}(\phi) \to \mathrm{Prf}(\psi)]$.

Finally, the internal language supports the axioms of guarded recursion (a later modality $\triangleright$, Löb induction, etc.). Surprisingly, this structure is not relevant to our discussion of resource algebras and so we refer the reader to Birkedal et al. [3] for details and an explanation of the semantics of the internal language.

2.2 Partial maps within $\mathbf{PSh}(\omega)$

For a first example of working within the internal language of $\mathbf{PSh}(\omega)$ and in preparation for Section 3, we define the notion of a partial map. A naïve first guess might be to say that a partial map from $A \rightharpoonup B$ consists of an ordinary function $A \to \mathbf{1} + B$. While this definition suffices in 'ordinary' external mathematics, it is too strict when working internally as we are. Indeed, this definition only captures functions whose domain of definition is *decidable* and we will frequently be interested in partial functions f where $f(a)\!\downarrow$ is an arbitrary proposition and therefore—since $\mathbf{PSh}(\omega)$ does not satisfy excluded middle—typically undecidable.

We instead use the *partial map classifier* following Rosilini [23]:

Definition 2. *If $A : \mathcal{U}$ then the partial map classifier* $A^? = \sum_{\phi : \mathrm{Prop}} \mathrm{Prf}(\phi) \to A$.

A map $f : A \to B^?$ encodes a partial map from A to B. Note that f consists of (1) a map $f_0 : A \to \mathrm{Prop}$ indicating where f is defined and (2) a map $f_1 : (a : A) \to \mathrm{Prf}(f_0\, a) \to B$ giving the value of f when it is defined.

Definition 3. *We define the type of partial maps $A \rightharpoonup B$ to be $A \to B^?$*

Lemma 1. *$-^?$ is a monad.*

We use monadic notation ($x \leftarrow a; b(x)$ and $\mathsf{ret}(x)$) to manipulate $A^?$ and write $f^? : A^? \to B^?$ for the functorial action.

Definition 4. *We define the approximation partial order $(\phi, a) \sqsubseteq (\psi, b) : \mathrm{Prop}$ on $A^?$ as $\exists f : \mathrm{Prf}(\phi) \to \mathrm{Prf}(\psi). (z : \mathrm{Prf}(\phi)) \to a\, z = b(f\, z)$.*

3 Resource Algebras and the Maximal Idempotent Axiom

We now analyze the abstract definition of resources using the internal language. While this definition is closely related to Iris's cameras, we avoid several maneuvers used by Iris to optimize for formalization in order to take full advantage of the internal language. A full comparison is provided in Section 6. We show how various portions of the definition may be replaced and folded into a single new axiom we term the *maximal idempotent axiom*.

3.1 Partial commutative semigroups and predicates

We begin with the basic notion of a type equipped with a partial multiplication:

Definition 5. *A partial commutative semigroup (PCS) is a type A and a partial map $(\cdot) : A \times A \rightharpoonup A$ satisfying the following conditions:*

1. *If $a, b : A$ then $a \cdot b = b \cdot a : A^?$*
2. *If $a, b, c : A$ then $x \leftarrow (a \cdot b); x \cdot c = x \leftarrow (b \cdot c); a \cdot x : A^?$*

In light of (1) and (2), we may unambiguously write $a \cdot b \cdot c$.

Lemma 2. *If A is a PCS then $a \leq b \triangleq a = b \vee \exists c : A.\, a \cdot c = \mathsf{ret}\, b$ is a preorder.*

As mentioned in the introduction, the semantics of separation logics like Iris can be constructed within monotone predicates on a PCS $(A, \cdot)$:

$$A \xrightarrow{\mathsf{mon}} \mathrm{Prop} \triangleq \{P : A \to \mathrm{Prop} \mid \forall a \leq b.\, P(a) \to P(b)\}$$

We will not spell out how all the connectives of higher-order logic are interpreted into $A \xrightarrow{\mathsf{mon}} \mathrm{Prop}$, but we record the definition of $*$ for intuition:

$$(P * Q)\, a \triangleq \exists a_0, a_1 : A, a_0 \cdot a_1 = \mathsf{ret}(a) \wedge P(a_0) \wedge Q(a_1)$$

3.2 Core structures

The classical definition of a camera is a PCS with a core operation:

Definition 6. *If $(A, \cdot)$ is a PCS, a core structure on A is a map $|-| : A \to \mathbf{1} + A$ satisfying the following conditions:*

1. *If $|a| = \mathsf{in}_2(a')$ then $a' \cdot a = \mathsf{ret}\, a$.*
2. *If $|a| = \mathsf{in}_2(a')$ then $|a'| = \mathsf{in}_2(a')$.*
3. *If $a \leq b$ then $|a| \leq |b|$*

For the last point, we order $\mathbf{1} + A$ such that $\mathsf{in}_1(\star)$ is the minimum element.

Definition 7. *A camera is a PCS equipped with a core structure.*[4]

As mentioned in the introduction, any core structure induces a comonadic modality $\Box$ on monotone predicates $A \xrightarrow{\mathsf{mon}} \mathrm{Prop}$:

$$\Box P \triangleq \lambda a \to \begin{cases} \bot & |a| = \mathsf{in}_1(\star) \\ P(a') & |a| = \mathsf{in}_2(a') \end{cases}$$

Lemma 3. *If $|-|$ is a core structure then the induced $\Box$ modality is monotone, comonadic ($\Box P \to P$ and $\Box\Box P \iff \Box P$), and commutes with $\exists$, $\triangleright$, and satisfies the following: $(\Box P) \wedge Q \iff (\Box P) * Q$*

For the purposes of separation logic, it is the last property that is most vital. It allows a user of separation logic to freely exchange separating and ordinary conjunction when manipulating $\Box P$ and thereby allows one to e.g., duplicate $\Box P$ into $\Box P * \Box P$ and similar. Intuitively, $\Box P$ is the closest replacement of P which satisfies this property. Unfortunately, this idea is complicated by the fact that there are often great many distinct core structures which one may choose to equip $(A, \cdot)$ with, and they induce distinct modalities. It is thus far from clear which core structure (if any) actually forces $\Box P$ to realize the above description.

This situation is in contrast to the motivating example with monotone predicates on heaps where the $\Box$ modality could be characterized by several universal properties. Our goal is to isolate a property of core which fully constrains its definition to give the strongest possible $\Box$ modality. That is, we wish to find a core structure $|-|$ so that if $|-|'$ is another core structure then $\forall P. \Box_{|-|'} P \to \Box_{|-|} P$.

We begin by massaging Definition 6 into a form which is easier to work with. If $|-|$ is a core structure and $|a| = \mathsf{in}_2(a')$ then a' is an idempotent element (by 1 and 2). In light of 3, we may therefore view $|-|$ as a monotone assignment $A \to \mathbf{1} + \mathsf{Idem}(A)$ and, more specifically, an assignment $(a : A) \to \{x : \mathbf{1} + \mathsf{Idem}(A) \mid x \leq \mathsf{in}_2(a)\}$.

Lemma 4. *A core structure on $(A, \cdot)$ is equivalent to a monotone map $(a : A) \to \{x : \mathbf{1} + \mathsf{Idem}(A) \mid x \leq \mathsf{in}_2(a)\}$.*

[4] Jung et al. [16], required an additional axiom to force $\triangleright$ to commute with $*$. We have followed e.g. Spies et al. [24] and removed this requirement to allow for more models.

A great many such monotone maps are possible for most PCSs. For instance, the constant map $\lambda a \to \mathsf{in}_1(\star)$ is always available and if A has a global unit ϵ, then $\lambda a \to \mathsf{in}_2(\epsilon)$ is also a valid choice. As core structures are particular monotone maps between preorders, they themselves inherit a preorder defined pointwise.

Lemma 5. *The assignment of core structures to $\Box$ modalities is monotone: if $|-|$ and $|-|'$ are two core structures such that $|-| \leq |-|'$ then $\Box_{|-|}P \to \Box_{|-|'}P$.*

We note that we have have already encountered the unique minimal core structure for any PCS: the assignment $\lambda a \to \mathsf{in}_1(\star)$ which discards all information of a. Unfolding, this core structure induces the $\Box$ modality which sends every proposition to $\bot$, which is clearly the smallest possible modality. Much more useful is the *maximal* $\Box$ modality.

In order to obtain the largest possible $\Box$ modality, it suffices to construct a core structure which is maximal. In particular, it would suffice to ensure $|-|$ sends a to the largest possible idempotent element contained within a and which is undefined only if no idempotent element exists. Notice that a priori there may be distinct maximal core structures as A is merely pre-ordered, but every PCS is partially-ordered on idempotent elements, in particular:

Lemma 6. *If i, j are idempotents such that $i \leq j$ and $j \leq i$ then $i = j$.*

Proof. By assumption so there exists i' and j' such that $i \cdot i' = \mathsf{ret}\, j$ and $j \cdot j' = \mathsf{ret}\, i$. Since both i and j are idempotent, this tells us that $j \cdot i = \mathsf{ret}\, i$ and $i \cdot j = \mathsf{ret}\, j$. Commutativity then tells us that $i = j$.

In particular, core structures are partially-ordered and (using pointwise multiplication) form a join semi-lattice. We therefore conclude the following:

Corollary 1. *Maximal core structures are unique.*

Unfortunately, there is no guarantee that a maximal core structure need exist:

Lemma 7. *There exist a PCS for which there is no maximal core structure.*

Proof. Consider the PCS on $A = \mathbb{N} + \mathbf{1} = \{0 \ldots \infty\}$ where (1) $n \cdot m = \max(n, m)$ (2) $n \cdot \infty = \infty \cdot n = \infty$ and (3) $\infty \cdot \infty$ is undefined. Note that $\mathsf{Idem}(A)$ consists of all elements of A except ∞.

To define a core structure on A, note that if $|\infty| = \mathsf{in}_2(n)$, then $|m| \leq \mathsf{in}_2(n)$ for all m by (3) and if $|\infty| = \mathsf{in}_1(\star)$ then $|m| = \mathsf{in}_1(\star)$. It follows that any core structure where $|\infty| = \mathsf{in}_2(n)$ must be smaller than the core structure where $|\infty|^{n+1} = \mathsf{in}_2(n+1)$ and $|m|^{n+1} = \mathsf{in}_2(\min(n+1, m))$. This yields an ascending chain of core structures $|-|^1 < |-|^2 \ldots$ for which there is no upper bound.

Fortunately, maximal core structures do exist in a wide variety of circumstances. For instance, it was necessary that our above counterexample involved an infinite number of idempotent elements.

Lemma 8. *If $\{x : \mathbf{1} + \mathsf{Idem}(A) \mid x \leq \mathsf{in}_2(a)\}$ is a decidable finite type for each $a : A$, then A has a maximal core structure.*

Corollary 2. Heap *has a maximal core structure: $\lambda h \to \epsilon$.*

3.3 The maximal idempotent axiom

We can reformulate the requirement that PCS possess a maximal core structure into the following property:

Definition 8 (The Maximal Idempotent Axiom). *A PCS $(A, \cdot)$ satisfies the maximal idempotent (MI) axiom if the following holds for all $a : A$:*

$$\mathsf{MI}(a) \triangleq \neg(\exists i \leq a.\, i\,\mathsf{idem}) \vee (\exists i \leq a.\, i\,\mathsf{idem} \wedge \forall j \leq a. j\,\mathsf{idem} \to j \leq i)$$

In other words, below each $a : A$ there either exists a maximal idempotent or there are no idempotents at all.

A priori, one may worry that the mere existence of a maximal idempotent below $a : A$ does not suffice to construct a function $|-|$ picking it out. After all, in general constructing such a function amounts to the axiom of choice and the lack of excluded middle implies the failure of the axiom of choice internal to $\mathbf{PSh}(\omega)$. However, as a topos $\mathbf{PSh}(\omega)$ does satisfy the axiom of *unique* choice: if $\forall a : A. \exists! b : B. \Phi(a, b)$ then $\exists! f : A \to B. \forall a : A. \Phi(a, f(a))$. Since maximal idempotents are unique, we obtain the following:

Theorem 1. *If $(A, \cdot)$ satisfies the MI axiom then A has a maximal core structure.*

Lemma 9. *If $\phi, \psi : \mathrm{Prop}$ such that $\phi \wedge \psi = \bot$ then $\mathrm{Prf}(\phi \vee \psi) \cong \mathrm{Prf}(\phi) + \mathrm{Prf}(\psi)$.*

Lemma 10. *If $A : \mathcal{U}$ and $\phi : A \to \mathrm{Prop}$ such that $\exists a : A.\, \phi\, z = \exists! a : A.\, \phi\, z$ then $\mathrm{Prf}(\exists a : A.\, \phi\, z) \cong \sum_{a:A} \mathrm{Prf}(\phi\, z)$.*

Proof (Theorem 1). Since the total absence of idempotents and the presence of maximal idempotent are disjoint, $\mathrm{Prf}(\mathsf{MI}(a))$ is equal to the following:

$$(\mathrm{Prf}(\exists i \leq a.\, i\,\mathsf{idem}) \to \bot) + \mathrm{Prf}(\exists i \leq a.\, i\,\mathsf{idem} \wedge \forall j \leq a. j\,\mathsf{idem} \to j \leq i)$$

Next, since maximal idempotents are unique, the second summand is equivalent to *unique* existence, whereby we may replace it with the following:

$$((i : A) \to i \leq a \to i\,\mathsf{idem} \to \bot) + \textstyle\sum_{i \leq a} i\,\mathsf{idem} \times ((j \leq a) \to j\,\mathsf{idem} \to j \leq i)$$

In other words, $\mathrm{Prf}(\forall a : A.\, \mathsf{MI}(a))$ is equivalent to a choice function sending each element of A to the maximal idempotent below A or a proof that there are no idempotents below A at all. This function is the required maximal core structure.

This addresses a rather mysterious fact about cameras: why are core structures partial in a different way than $\cdot$? In light of the above, we see that a core structure $|-|$ has decidable support because $|a|$ is undefined just when a contains *no* idempotents at all and the MI axiom guarantees that this is decidable.

We now *redefine* and simplify cameras from their formulation in Iris. Rather than insisting that a camera be a PCS equipped with a core structure, we instead ask that it satisfy the MI axiom. This has the simultaneous advantages of (1) ensuring that the induced core structure is as strong as possible while also (2) merely being a property to check, not an additional structure.

Definition 9. *A resource algebra is a PCS satisfying the maximal idempotent axiom. Write* RA *for the type of all resource algebras.*

To justify this definition, we show that it is not overly restrictive (Section 4) and that it opens up more conceptual descriptions of existing cameras (Section 5).

4 Examples of cameras

If Definition 9 is to be an adequate replacement for the current definition of cameras in Iris, it must not rule out examples which are currently used in practice. Fortunately, every camera currently used in the Iris formalization in Coq satisfies the MI axiom. For reasons of space, we present only a handful of the most important resource algebras used in separation logic and illustrate the proofs that they satisfy the MI axiom. We divide these examples into two categories: basic stock cameras which are used in a wide variety of constructions and a few operations on cameras used to build up more complex examples.

4.1 Basic resource algebras

For the basic 'building block' resource algebras, Lemma 8 suffices to show that the relevant PCSs satisfy the MI axiom; in practice, they all have no idempotent elements, one idempotent element, or every element is idempotent. We illustrate this with two representative examples.

First, the *exclusive resource algebra* which is used to encode ownership of non-duplicable abstract resources:

Lemma 11. *If* $A : \mathcal{U}$*, regard* A *as a PCS* $\mathsf{Excl}(A)$ *where* $\cdot_{\mathsf{Excl}(A)}$ *is undefined everywhere.* $\mathsf{Excl}(A)$ *satisfies the MI axiom.*

Proof. Such a camera has *no* idempotents, so Lemma 8 trivially applies.

For another extreme example, we consider the *agreement* resource algebra which is used to enforce global agreement about a particular value. For this, take a type $A : \mathcal{U}$ and regard A as a PCS $\mathsf{Ag}(A)$ using the following partial multiplication operation: $a \cdot_{\mathsf{Ag}(A)} a' \triangleq (a = a', \lambda\_ \to a)$.

Lemma 12. *For any* $A : \mathcal{U}$*,* $\mathsf{Ag}(A)$ *satisfies the MI axiom.*

Proof. Lemma 8 applies as the only element contained in $\mathsf{Idem}(A)_{\leq a}$ is a itself.

4.2 Constructions on resource algebras

The power of resource algebras is their composability: given simple resource algebras like those outlined above, we may stitch them together into more powerful constructions like Heap. We outline the most important constructions for combining resource algebras and show that they preserve the MI axiom.

Given two resource algebras A and B we consider pointwise multiplication:

$$(a, b) \cdot_{A \times B} (a', b') \triangleq \bar{a} \leftarrow a \cdot a'; \bar{b} \leftarrow b \cdot b'; \mathsf{ret}(\bar{a}, \bar{b})$$

Lemma 13. *If A and B are resource algebras, $A \times B$ satisfies the MI axiom.*

Proof. Fix a pair (a, b). An idempotent below (a, b) consists of a pair (i, j) of idempotents i below a and j below b. By the MI axiom of both A and B, it is decidable whether or not there exists any idempotents below either a or b whence whether any such pair (i, j) exists. By a further application of the MI axiom, if such a pair exists, there is a pair of maximal idempotents as required.

Given a resource algebra A, we can construct the resource algebra $\mathsf{Opt}(A) = \mathbf{1} + A$ adjoining a new element to A. This element will act as the unit of multiplication, meaning the partial multiplication operation will be defined as:

$$x \cdot_{\mathsf{Opt}(A)} y \triangleq \begin{cases} \mathsf{ret}(y) & x = \mathsf{in}_1(\star) \\ \mathsf{ret}(x) & y = \mathsf{in}_1(\star) \\ \mathsf{in}_2{}^?(a \cdot_A a') & x = \mathsf{in}_2(a), y = \mathsf{in}_2(a') \end{cases}$$

Lemma 14. $\mathsf{Opt}(A)$ *satisfies the MI axiom.*

Proof. Fix $x : \mathsf{Opt}(A)$. If $x = \mathsf{in}_1(\star)$, then x itself is idempotent and thus the necessary maximal idempotent. If instead $x = \mathsf{in}_2(a)$, then the idempotents below x are either of the form $\mathsf{in}_1(\star)$ or $\mathsf{in}_2(i)$ where i is an idempotent below a. Accordingly, the maximal idempotent below $\mathsf{in}_2(a)$ is either the maximal idempotent below a or $\mathsf{in}_1(\star)$ if no such idempotent exists.

For our final example, we describe a more complex construction assembling a collection of resource algebras together. This 'direct sum' operation is used pervasively in Iris: any real use of Iris is based on this resource algebra to stitch together all of the resources required the verification of a given program. We describe it in more generality than it is presented in Iris by allowing for direct sums to be indexed by a type I with decidable equality.

Fix a family of resource algebras $A : (i : I) \to \mathsf{RA}$, let $\bigoplus_{i:I} A\, i$ denote the type of partial maps $(i : I) \to A(i)$ with decidable and non-empty, finite support. Representing these in type theory can be somewhat difficult, but our internal language is strong enough to define quotient types (using Prop). Accordingly, we realize $\bigoplus_{i:I} A\, i$ as a quotient of the following type:

$$\textstyle\sum_{n:\mathbb{N}} \sum_{f:\mathbb{N}_{\leq n} \hookrightarrow I} (n : \mathbb{N}_{\leq n}) \to A(f\, n)$$

The quotienting relation identifies two triples (n, f, a) and (n', f', a') just when $n = n'$ and there is a bijection $\sigma : \mathbb{N}_{\leq n} \to \mathbb{N}_{\leq n}$ such that $f = f' \circ \sigma$ and $a = a' \circ \sigma$.

For convenience, we will write elements of $\bigoplus_{i:I} A\, i$ as $\{(i_0, a_0), \ldots, (i_n, a_n)\}$ where $i_0, \ldots, i_n$ are disjoint elements of I and $a_k : A\, i_k$.

Lemma 15. $\bigoplus_{i:I} A\, i$ *is a partial semigroup under pointwise multiplication.*

Proof (Sketch). The definition of pointwise multiplication is somewhat involved and so we only sketch it here. Fix two elements $\rho = \{(i_0, a_0), \ldots, (i_n, a_n)\}$ and $\rho' = \{(j_0, b_0), \ldots, (j_m, b_m)\}$. Rearranging the elements of ρ' if necessary, we

assume that there exists ℓ such that $i_k = j_k$ if $k < \ell$ and outside of these pairs, the sequences Is are disjoint. Note that this requires I to have decidable equality.

We define the $\rho \cdot \rho'$ by multiplying together the ℓ elements overlapping between ρ and ρ' and, when these are all defined, returning the partial map with these the results and along with the elements of ρ and ρ' outside the overlap:

$$\begin{aligned}
&\rho \cdot \rho' \triangleq \\
&\quad c_0 \leftarrow a_0 \cdot_{A\, i_0} b_0; \\
&\quad \dots \\
&\quad c_{\ell-1} \leftarrow a_{\ell-1} \cdot_{A(\ell-1)} b_{\ell-1}; \\
&\quad \mathsf{ret}\, \{(i_0, c_0), \dots (i_{\ell-1}, c_{\ell-1}), (i_\ell, a_\ell) \dots (i_n, a_n), (j_\ell, b_\ell) \dots (j_m, b_m)\}
\end{aligned}$$

Since $-^?$ is a commutative monad, the order of these multiplications are irrelevant and so this process respects the equivalence relation quotienting $\bigoplus_{i:I} A\, i$.

Theorem 2. *$\bigoplus_{i:I} A\, i$ satisfies the MI axiom.*

Proof. Fix $\rho = \{(i_0, a_0), \dots, (i_n, a_n)\}$. Since $\mathsf{MI}(\rho)$ is a proposition the quotienting is irrelevant and we ignore it. Next, note that if $\sigma \cdot \sigma' = \mathsf{ret}(\rho)$, then the support of σ and σ' must be subsets of the support of ρ. Accordingly, if σ is an idempotent within ρ, then it consists of a collection of idempotents drawn from $A\, i_k$ below the corresponding element a_k.

Since each $A\, i_k$ is a resource algebra, the MI axiom tells us that there is either (1) no idempotent below a_k or (2) a maximal idempotent a'_k below a_k. Without loss of generality, let us assume that a_k has a maximal idempotent if and only if $k \leq m$ for some m and no idempotent below it $k > m$.

The maximal idempotent below ρ is then given by $\sigma = \{(i_0, a'_0), \dots, (i_m, a'_m)\}$. Indeed, any other idempotent σ' below ρ must have a smaller support than σ and, where both are defined, the value of σ' smaller than that of σ by construction.

Note that the above depends critically on the fact that the existence of a maximal idempotent is *decidable*. We must be able to analyze whether the i_k entry is to be included based on whether or not a_k contains a maximal idempotent.

To conclude this section, we describe how one can combine these simpler resource algebras into more complex constructions needed for realistic program logics. For instance, the foundational resource algebra representing heaps can be constructed as follows. Writing Loc and Val to be the types of locations and (syntactic) values in a hypothetical imperative language, we define the resource algebra of heaps in this language as follows:

$$\mathsf{Heap} \triangleq \mathsf{Opt}\big(\textstyle\bigoplus_{l:\mathsf{Loc}} \mathsf{Excl}(\mathsf{Val})\big)$$

In this definition $\mathsf{in}_1(\star)$ represents the empty heap, and the exclusive resouce algebra ensures that multiplication of heaps is only defined when the locations of the relevant heaps are disjoint. In particular, there is no need to check the MI axiom here: it is automatic since all constructions involved preserve its validity.

5 The Category of Resource Algebras

We now turn to the properties of resource algebras *collectively* by structuring them into a category. In particular, we show that many resource algebras that are important in practice satisfy simple and recognizable universal properties. The proofs of these facts are not terribly difficult, but this is the point; by isolating a well-behaved definition of resource algebra these calculations are of the sort familiar to any category of algebraic structures.

To define the category of resource algebras, we must, of course, decide on what a morphism of resource algebras should be. Since a resource algebra is a PCS satisfying an additional property, a first idea is to define the category of resource algebras to be a full subcategory of partial commutative semigroups. This is analogous to how, e.g., a morphism of abelian groups is an ordinary group homomorphism whose domain and codomain happen to be abelian. When formulating morphisms of PCSs, the real wrinkle is partiality; from a categorical point of view, PCSs are commutative semigroups in the *Kleisli category* of $-^?$. But Kleisli categories for a monad are usually poorly behaved and $-^?$ is no exception. The solution is to integrate the partial ordering $\sqsubseteq$ introduced in Section 2:

Definition 10. *A* morphism of resource algebras $f : (A, \cdot) \to (B, \cdot)$ *is a function* $f : A \to B$ *such that* $f^?(a_0 \cdot a_1) \sqsubseteq f(a_0) \cdot f(a_1)$ *when* $a_0, a_1 : A$.

Informally: a morphism of resource algebras is a morphism between carriers which commutes with multiplication *provided the multiplication is defined in the domain*. Such morphisms are commonly referred to as *lax* morphisms since we do not, e.g., require $f(a_0) \cdot f(a_1)$ to be *undefined* if $a_0 \cdot a_1$ is undefined. As we shall see, lax morphisms enjoy better categorical properties.

Lemma 16. *Resource algebras and morphisms of such form a category* **RA**.

5.1 First steps with RA

We now record a few basic properties of **RA**:

Lemma 17. RA *has finite products and finite coproducts.*

Proof. Initial and terminal objects are given by **0** and **1** with the obvious multiplication maps. We have already encountered Lemma 13 and, since the construction of coproducts is unsurprising, we content ourselves with showing that $A \times B$ has the expected universal property.

To this end, observe that the projection maps $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ are both morphisms of resource algebras. Surprisingly, this is the most subtle part of the proof and it depends upon the fact that we require only that ${\pi_1}^?(p \cdot q) \sqsubseteq \pi_1\, p \cdot \pi_p\, q$ rather than equality. To see that the universal property is satisfied, we need only show that if C is a resource algebra such that $f : C \to A$ and $g : C \to B$ are morphisms, then $\lambda c.\, (f\, c, g\, c)$ is a morphism to $A \times B$. This is a routine calculation.

The universal property of the direct sum algebra is slightly longer to describe as it is not a simple limit or colimit. However, in light of the complexity of the definition of multiplication in $\bigoplus_{i:I} A\,i$, we note that even though the universal property is complex, it is significantly more conceptual than the actual definition.

Definition 11. *A compatible cocone over A is a family of morphisms $(f_i : A\,i \to C)_{i:I}$ such that for every finite family of elements $a_0 : A\,i_0, \dots, a_n : A\,i_n$ the product $f_{i_0}\,a_0 \cdot \ldots \cdot f_{i_n}\,a_n$ is defined.*

Lemma 18. *The resource algebra $\bigoplus_{i:I} A\,i$ along with the canonical inclusions $A\,i \to \bigoplus_{i:I} A\,i$ is the universal compatible cocone over A.*

In other words, modulo partiality $\bigoplus_{i:I} A\,i$ is the coproduct of A; it is the universal way to include each $A\,i$ inside a single resource algebra in such a way that multiplications between disjoint elements is permitted.

5.2 The relation between RA and other categories

A number of the resource algebras encountered in Section 4 can be characterized as adjoints of natural functors between **RA** and other categories. For instance, Excl and Ag are both left adjoints to certain functors from **RA** to the category of small types and functions between them (denoted **TY**).

Lemma 19. Excl *is left adjoint to the forgetful functor* $\mathbf{RA} \to \mathbf{TY}$.

Proof. Examining definitions, since multiplication in Excl is always undefined, a morphism $\mathsf{Excl}(A) \to B$ is precisely a map from A to the carrier of B, as required.

A similar argument gives a universal characterization of Ag:

Lemma 20. Ag *is left adjoint to the functor* $\mathsf{Idem} : \mathbf{RA} \to \mathbf{TY}$ *that maps a resource algebra to its set of idempotents.*

The final result concerns Opt. As this resource algebra extends an existing resource algebra with a unit element (i.e., an element ϵ where $\epsilon \cdot - = \mathsf{ret}$), it is reasonable to guess that $\mathsf{Opt}(A)$ is the resource algebra "freely generated" by A along with a unit. To state this precisely, we introduce *unital* resource algebras:

Definition 12. *A* unital resource algebra *is a resource algebra $(A, \cdot)$ with a unit and a unital morphism a morphism of resource algebras preserving the unit.*

Definition 13. *The category of unital resource algebras is denoted* **uRA**.

Lemma 21. *There is a forgetful functor $U : \mathbf{uRA} \to \mathbf{RA}$ and the left adjoint to this functor is given by* Opt.

We note that while the proof of this lemma is straightforward, it is simply *false* if the definition of resource algebra included a core structure as data.

For a small example of how all of these universal properties combine, let us turn to the 'map of resource algebras' used to instantiate Iris: $\mathsf{Opt}\big(\bigoplus_i A\,i\big)$. We can give a concise description of this rather large resource algebra as the universal method of collecting together the family of resource algebras A and freely adjoining a unit.

6 From the topos of trees to the Iris formalization

Thus far, we have taken our motivation from Iris and its formalization in Coq but focused our attention on the better-behaved $\mathbf{PSh}(\omega)$ and its internal type theory. While this is ideal for working on paper, Iris and its realization in Coq must also optimize for the practicalities of formalization with Coq and therefore use only a subcategory of $\mathbf{PSh}(\omega)$ and a more ad-hoc replacement for $-^?$ in its definition of cameras. In this section, we discuss the impact of our results in Sections 3 to 5 for Iris and its formalization in Coq. Within this section we disregard the internal language of $\mathbf{PSh}(\omega)$ and work externally. For clarity, we refer to cameras as they appear in the Iris formalization as *Iris cameras* and reserve the terms camera and resource algebra for the notions discussed in Section 3.

6.1 Cameras in Iris

We begin with a brief summary of Iris. Rather than using all of $\mathbf{PSh}(\omega)$, Iris's basic types are drawn from the full subcategory of *total presheaves*:

Definition 14. *A total presheaf $X : \mathbf{PSh}(\omega)$ is one whose restriction maps $X(n+1) \to X(n)$ are surjective.*

Equivalently, a total presheaf is a set X together with a family of equivalence relations $(\equiv_0) \subseteq (\equiv_1) \dots$ such that $X = \mathsf{lim}_n\, X/(\equiv_n)$. Under this encoding, a natural transformation between total presheaves is a map of sets which respects all the equivalence relations. Total presheaves are closed under products, sums, exponentials, the partial map classifier, and various other operations in $\mathbf{PSh}(\omega)$. Notably, however, they are not closed under finite limits, do not form a locally cartesian closed category, and lack a strong internal language.

To ease formalization, Iris avoids using $-^?$ and encodes of partiality indirectly:

Definition 15. *A (total)* lifted *PCS (LPCS) $(X, \cdot, \mathcal{V})$ is a (total) presheaf X with an associative and commutative natural transformation $\cdot : X \times X \to X$ and a* validity predicate $\mathcal{V} : X \to \mathrm{Prop}$ *such that $\forall x, y \in X(n).\, \mathcal{V}_n(x \cdot_n y) \subseteq \mathcal{V}_n(x)$.*[5]

Intuitively, we have arranged for $\cdot$ to be total by having X also contain sentinel values. The role of $\mathcal{V}$ is then to carve out those elements of X which are genuine from those merely representing undefined multiplications.

One PCS $(M, \cdot)$ induces an LPCS by taking $(M^?, \bar{\cdot}, \pi_1)$ where $\bar{\cdot} : M^? \times M^? \to M^?$ is the Kleisli extension of $\cdot$. The reverse is also possible: given $(X, \cdot, \mathcal{V})$ one can take the subpresheaf of valid elements along with the partial map induced by $\cdot$ defined only when $x \cdot y$ is valid. The loop PCS $\to$ LPCS $\to$ PCS is the identity.

Definition 16. *A core structure on a total LPCS $(X, \cdot, \mathcal{V})$ is a natural transformation $|-| : X \to \mathbf{1} + X$ satisfying the following properties:*

- *If $x \in X(n)$ and $|x| = \mathsf{in}_2(x')$ then $x' \cdot x = x$ and $|x'| = \mathsf{in}_2(x')$.*

[5] Recall that in $\mathbf{PSh}(\omega)$ the set $\mathrm{Prop}(n)$ is given by down-closed sets of $\{0 \dots n\}$.

– $|-|$ *is monotone with respect to the extension order on* X.

In general, a core structure on an LPCS is distinct from a core structure on the induced PCS due to the presence of invalid elements. However, one can always extend a core structure on a PCS to a core structure on the induced LPCS.

Definition 17. *An Iris camera is a total LPCS equipped with a core structure.*

In fact, the definition is more relaxed: a total presheaf can be represented as a family of equivalence relations on a set $(X, (\equiv)_n)$ satisfying $X = \mathsf{lim}_n\, X/(\equiv)_n$. Iris currently requires only the strictly weaker condition that $(\bigcap_n (\equiv_n)) = \{(x, x) \mid x \in X\}$ of its cameras to e.g., give a particular definition of the agreement camera. We will return to this point in Section 6.3.

6.2 Maximal idempotent axiom for Iris cameras

Despite the difference between a core structure on an Iris camera and a core structure on the induced partial commutative semigroup, one can translate the definition of the MI axiom into the language Iris cameras:

Definition 18. *An Iris camera* X *satisfies the maximal idempotent axiom if the following holds for every* n *and* $x : X(n)$ *such that* $n \in \mathcal{V}_n(x)$*:*

$$(\forall y : X(0).\, y \leq x|_0 \land y \cdot_0 y = y \to \bot)$$
$$\lor\, (\exists y : X(n).\, y \leq x \land y \cdot_n y = y \land \forall m \leq n, z : X(m).\, z \leq x|_m \land z \cdot_m z = z \to z \leq y|_m)$$

We have written e.g., $x|_m$ *for the functorial action* $X(m \leq n)(x)$*.*

This definition is mechanically produced by unfolding the internal definition in $\mathbf{PSh}(\omega)$ of the maximal idempotent axiom [3]. For instance, PCS satisfies the MI axiom if and only if the induced LPCS satisfies the above proposition.

Theorem 3. *Every Iris camera in the Iris formalization satisfies the MI axiom.*

We have proven this by extending the Iris formalization with proofs of the MI axiom for every Iris camera. Consequently, we are able to *remove* the core structures currently defined in the Iris formalization and replace them with the induced maximal core structure. This change does not disrupt any other portion of the formalization. In total then, the theoretical work done in Section 3 has allowed us to remove explicit constructions in the Iris formalization and replace them with more conceptual arguments all without requiring any change to the complicated proofs in separation logic currently carried out within Iris.

Remark 1. As discussed in Section 3, one may as well replace $\lor$ with $+$ and $\exists$ with $\sum$. When extending the Iris formalization, we have made these substitutions as Coq's metatheory is too weak to justify either replacement without axioms.

6.3 Categories of Iris cameras

While the MI axiom can be used to save effort and complexity in the formalization, the results of Section 5 do not translate so easily. There are several incompatible possible definitions of morphisms for Iris cameras and none are fully satisfactory. For instance, morphisms may be themselves total or only defined on valid elements, they may preserve validity laxly or strictly, etc. Each of these definitions gives rise to a category satisfying some, but not all, of the properties of **RA**. Fundamentally, the issue is that totalizing a PCS forces us to consider invalid elements.

There exists a well-behaved subclass of LPCSs which organize into a category: the image of the map $(M, \cdot) \mapsto (M^?, \bar{\cdot}, \pi_1)$. However, important Iris cameras do not land in this class of LPCSs because e.g., their underlying types are discrete.

For a concrete example of how this state of affairs complicates research in Iris, consider the resource algebra $\mathsf{Ag}(A)$ from Section 4. It induces an Iris camera by passing to the induced LPCS $\hat{\mathsf{Ag}}(A) = \mathsf{Ag}(A)^?$ which satisfies the desiderata of the agreement Iris camera [16, Section 4.3]. A closely related construction appeared in Jung et al. [15], but was criticized for being exceptionally difficult to define [16]; this suggests an advantage of our categorical approach which yields $\hat{\mathsf{Ag}}(A)$ as the composition of a number of simple and canonical results.

Rather than $\hat{\mathsf{Ag}}(A)$, Jung et al. [16] use a more ad-hoc construction $\mathsf{Ag}'(A)$. This construction is easier to formalize, but $\mathsf{Ag}'(A)$ is *not* a total presheaf: we do not have $\mathsf{Ag}'(A) = \mathsf{lim}_n \mathsf{Ag}'(A)/(\equiv)_n$ because of the presence of 'junk' elements which are never valid. This motivated practitioners to stop requiring that Iris cameras satisfy the completeness condition. However, these junk elements ensure that $\mathsf{Ag}'(A)$ is not the totalization of a PCS and it is therefore not among those LPCSs which organize into a well-behaved category. Consequently, we have no good universal property describing $\mathsf{Ag}'(A)$.

From our point of view, however, the issue is not with $\hat{\mathsf{Ag}}(A)$—it has the expected universal property—but with the insistence on lifted PCSs. For instance, while $\mathsf{Ag}'(A)$ is easier to deal with than $\mathsf{Ag}(A)^?$, it is still more complex than the underlying type of $\mathsf{Ag}(A)$: A! Strikingly, $\mathsf{Ag}(A)$ is simpler to define than $\mathsf{Ag}'(A)$ and the motivating technical advantage recommending $\mathsf{Ag}'(A)$ over $\hat{\mathsf{Ag}}(A)$—that it preserves 'timelessness'—follows without any additional effort for the resource algebra $\mathsf{Ag}(A)$. More generally, if one works with resource algebras instead of Iris cameras, the underlying types are simplified, the multiplication operation is more transparent, and the results enjoy universal properties.

Fortunately, most of the payoff of our investigation of resource algebras is contained in Theorem 3; for formalization purposes, it is not necessary to have a well-behaved category of Iris cameras, even if one is useful for discovering and explaining particular examples. We therefore view Section 5 as evidence that future iterations of Iris and similar separation logics would benefit from using a (1) broader class of presheaves and (2) using PCSs over LPCSs.

7 Related Work

A great deal of effort has focused on the logical underpinnings of (concurrent) separation logics and our work fits into this tradition. The logical core of separation logic was developed under the name bunched implication logic [13, 21] and subsequently generalized by Biering et al. [2] to account for higher-order separation logics. The importance of partial commutative monoids in separation logic was apparent in these models and PCMs were subsequently featured prominently in concurrent separation logics [6, 8, 9, 14]. Early versions of Iris [17] also included partial commutative monoids, but later replaced them with cameras [15, 16, 18].

Directly related is the work by Bizjak and Birkedal [4] which analyzes the behavior of $\Box$ modalities in terms of the core structures which induce them. They show that there is a bijection between $\Box$ modalities and core structures in the non-step-indexed setting and a weaker correspondence in a constructive setting. We have used this as inspiration for our approach to uniquely *define* core as the operation inducing the largest possible $\Box$ modality. They also elucidate the connection between $\mathbf{PSh}(\omega)$ and the model of Iris [16].

Finally, we note that our observation that total presheaves can be limiting when extending the model of Iris was also noted by Spies et al. [24] who found several technical issues generalizing total presheaves to a transfinite setting.

8 Conclusion and Future Work

A key aspect of modern separation logics is how resources are modeled. In this paper we have introduced a novel notion of *resource algebra*, which differs in a subtle but crucial way from the camera definition used in Iris, by omitting the core operation and replacing it by a property, the maximal idempotent axiom. We have demonstrated that the new definition improves upon the earlier one in the sense that it induces a universal $\Box$ modality and a well-behaved category RA of resource algebras. Moreover, we have shown that many of the known resource algebra constructions from Iris can be adapted to our new definition of resource algebra and that they satisfy universal properties. Finally, we have also shown that all the Iris cameras satisfy the maximal idempotent axiom.

Future work includes extending the Coq implementation of Iris to use the full category of presheaves rather than just a subcategory thereof, and thence to adopt our notion of resource algebra to such an implementation.

References

[1] Appel, A.W., McAllester, D.: An indexed model of recursive types for foundational proof-carrying code. ACM Transactions on Programming Languages and Systems 23(5), 657–683 (2001)

[2] Biering, B., Birkedal, L., Torp-Smith, N.: Bi-hyperdoctrines, higher-order separation logic, and abstraction. ACM Trans. Program. Lang. Syst. 29(5), 24–es (Aug 2007), https://doi.org/10.1145/1275497.1275499

[3] Birkedal, L., Møgelberg, R., Schwinghammer, J., Støvring, K.: First steps in synthetic guarded domain theory: step-indexing in the topos of trees. Logical Methods in Computer Science 8(4) (2012)
[4] Bizjak, A., Birkedal, L.: On models of higher-order separation logic. Electronic Notes in Theoretical Computer Science 336, 57–78 (04 2018), http://dx.doi.org/10.1016/j.entcs.2018.03.016
[5] Bornat, R., Calcagno, C., O'Hearn, P.W., Parkinson, M.J.: Permission accounting in separation logic. In: Palsberg, J., Abadi, M. (eds.) Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, January 12-14, 2005. pp. 259–270. ACM (2005), https://doi.org/10.1145/1040305.1040327
[6] Calcagno, C., O'Hearn, P.W., Yang, H.: Local action and abstract separation logic. In: 22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings. pp. 366–378. IEEE Computer Society (2007), https://doi.org/10.1109/LICS.2007.30
[7] Day, B.: On closed categories of functors. In: MacLane, S., Applegate, H., Barr, M., Day, B., Dubuc, E., Phreilambud, Pultr, A., Street, R., Tierney, M., Swierczkowski, S. (eds.) Reports of the Midwest Category Seminar IV. pp. 1–38. Springer Berlin Heidelberg, Berlin, Heidelberg (1970)
[8] Dinsdale-Young, T., Birkedal, L., Gardner, P., Parkinson, M.J., Yang, H.: Views: compositional reasoning for concurrent programs. In: Giacobazzi, R., Cousot, R. (eds.) The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013. pp. 287–300. ACM (2013), https://doi.org/10.1145/2429069.2429104
[9] Dockins, R., Hobor, A., Appel, A.W.: A fresh look at separation algebras and share accounting. In: Hu, Z. (ed.) Programming Languages and Systems, 7th Asian Symposium, APLAS 2009, Seoul, Korea, December 14-16, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5904, pp. 161–177. Springer (2009), https://doi.org/10.1007/978-3-642-10672-9_13
[10] Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. Logical Methods in Computer Science Volume 7, Issue 2 (06 2011)
[11] Hofmann, M.: Syntax and Semantics of Dependent Types. In: Pitts, A.M., Dybjer, P. (eds.) Semantics and Logics of Computation, pp. 79–130. Cambridge University Press (1997), https://www.tcs.ifi.lmu.de/mitarbeiter/martin-hofmann/pdfs/syntaxandsemanticsof-dependenttypes.pdf
[12] Hofmann, M., Streicher, T.: Lifting Grothendieck universes (1997), https://www2.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf, unpublished note
[13] Ishtiaq, S.S., O'Hearn, P.W.: Bi as an assertion language for mutable data structures. In: Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. POPL01, vol. 7, p. 14–26. ACM (01 2001), http://dx.doi.org/10.1145/360204.375719
[14] Jensen, J.B., Birkedal, L.: Fictional separation logic. In: Seidl, H. (ed.) Programming Languages and Systems - 21st European Symposium on Program-

ming, ESOP 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7211, pp. 377–396. Springer (2012), https://doi.org/10.1007/978-3-642-28869-2_19

[15] Jung, R., Krebbers, R., Birkedal, L., Dreyer, D.: Higher-order ghost state. SIGPLAN Not. 51(9), 256–269 (09 2016)

[16] Jung, R., Krebbers, R., Jourdan, J.H., Bizjak, A., Birkedal, L., Dreyer, D.: Iris from the ground up: A modular foundation for higher-order concurrent separation logic. Journal of Functional Programming 28 (2018)

[17] Jung, R., Swasey, D., Sieczkowski, F., Svendsen, K., Turon, A., Birkedal, L., Dreyer, D.: Iris: Monoids and invariants as an orthogonal basis for concurrent reasoning. SIGPLAN Not. 50(1), 637–650 (01 2015)

[18] Krebbers, R., Jung, R., Bizjak, A., Jourdan, J.H., Dreyer, D., Birkedal, L.: The Essence of Higher-Order Concurrent Separation Logic, p. 696–723. Springer Berlin Heidelberg (2017), http://dx.doi.org/10.1007/978-3-662-54434-1_26

[19] Mac Lane, S., Moerdijk, I.: Sheaves in geometry and logic : a first introduction to topos theory. Universitext, Springer (1992)

[20] O'Hearn, P., Reynolds, J., Yang, H.: Local Reasoning about Programs that Alter Data Structures, p. 1–19. Springer Berlin Heidelberg (2001), http://dx.doi.org/10.1007/3-540-44802-0_1

[21] O'Hearn, P.W., Pym, D.J.: The logic of bunched implications. Bulletin of Symbolic Logic 5(2), 215–244 (06 1999), http://dx.doi.org/10.2307/421090

[22] Reynolds, J.: Separation logic: a logic for shared mutable data structures. In: Proceedings 17th Annual IEEE Symposium on Logic in Computer Science (2002)

[23] Rosilini, G.: Continuity and effectiveness in topoi. Ph.D. thesis, University of Oxford (1986)

[24] Spies, S., Gäher, L., Gratzer, D., Tassarotti, J., Krebbers, R., Dreyer, D., Birkedal, L.: Transfinite Iris: Resolving an Existential Dilemma of Step-Indexed Separation Logic, p. 80–95. Association for Computing Machinery, New York, NY, USA (2021), https://doi.org/10.1145/3453483.3454031