Join Inverse Categories and Reversible Recursion[☆]

Robin Kaarsgaard[a,*], Holger Bock Axelsen[a], Robert Glück[a]

[a] *DIKU, Department of Computer Science, University of Copenhagen*

Abstract

Recently, a number of reversible functional programming languages have been proposed. Common to several of these is the assumption of totality, a property that is not necessarily desirable, and certainly not required in order to guarantee reversibility. In a categorical setting, however, faithfully capturing partiality requires handling it as additional structure. Recently, Giles studied inverse categories as a model of partial reversible (functional) programming. In this paper, we show how additionally assuming the existence of countable joins on such inverse categories leads to a number of properties that are desirable when modelling reversible functional programming, notably morphism schemes for reversible recursion, a †-trace, and algebraic ω-compactness. This gives a categorical account of reversible recursion, and, for the latter, provides an answer to the problem posed by Giles regarding the formulation of recursive data types at the inverse category level.

Keywords: reversible computing, recursion, categorical semantics, enriched category theory

1. Introduction

Reversible computing, that is, the study of computations that exhibit both forward and backward determinism, originally grew out of the thermodynamics of computation. Landauer's principle states that computations performed by some physical system (thermodynamically) dissipate heat when information is erased, but that no dissipation is entailed by information-preserving computations [3]. This has motivated a long study of diverse reversible computation models, such as logic circuits [4], Turing machines [5, 6], and many forms of restricted automata models [7, 8]. Reversibility concepts are important in quantum computing, but are increasingly seen to be of interest in other areas as well, including high-performance computing [9], process calculi [10], and even robotics [11, 12].

In this paper we concern ourselves with the categorical underpinnings of reversible functional programming languages. At the programming language level, reversible languages exhibit interesting program properties, such as easy program inversion [13]. Now, most reversible languages are stateful, giving them a fairly straightforward semantic interpretation [14]. While functional programs are usually easier to reason about at the meta-level, they do not have the concept of state that imperative languages do, making their semantics interesting objects of study.

Further, many reversible functional programming languages (such as Theseus [15] and the Π-family of combinator calculi [16]) come equipped with a tacit assumption of totality, a property that is neither required [6] nor necessarily desirable as far as guaranteeing reversibility is concerned. Shedding ourselves of the "tyranny of totality," however, requires us to handle partiality explicitly as additional categorical structure.

[☆] This is an extended version of an abstract presented at NWPT 2015 [1] and a paper presented at FoSSaCS 2016 [2], elaborated with full proofs, additional examples, and more comprehensive background.

[*] Corresponding author.

Email addresses: robin@di.ku.dk (Robin Kaarsgaard), funkstar@di.ku.dk (Holger Bock Axelsen), glueck@di.ku.dk (Robert Glück)

One approach which does precisely that is inverse categories, as studied by Cockett & Lack [17] as a specialization of restriction categories, which have recently been suggested and developed by Giles [18] as models of reversible (functional) programming. In this paper, we will argue that assuming ever slightly more structure on these inverse categories, namely the presence of *countable joins* of parallel morphisms [19], gives rise to a number of additional properties useful for modelling reversible functional programming. Notably, we obtain two different notions of reversible recursion (exemplified in the two different reversible languages RFUN and Theseus), and an account of recursive data types (via algebraic ω-compactness with respect to structure-preserving functors), which are not present in the general case. This is done by adopting two different, but complementary, views on inverse categories with countable joins as enriched categories – as **DCPO**-categories, and as (specifically Σ**Mon**-enriched) strong unique decomposition categories [20, 21].

Overview. We give a brief introduction to reversible functional programming, specifically to the languages of RFUN [22] and Theseus [15], in Section 2, and present the necessary background on restriction and inverse categories in Section 3. In Section 4 we show that inverse categories with countable joins are **DCPO**-enriched, which allows us to demonstrate the existence of (reversible!) fixed points of both morphism schemes and structure-preserving functors. In Section 5 we show that inverse categories with countable joins and a join-preserving disjointness tensor are (strong) unique decomposition categories equipped with a uniform †-trace. Section 6 gives conclusions and directions for future work.

2. On reversible functional programming

In this section, we give a brief introduction to reversible functional programming, specifically to the languages of RFUN and Theseus. For more comprehensive accounts of these languages, including syntax, semantics, program inversion, further examples, and so on, see [22] respectively [15].

Reversible programming deals with the construction of programs that are deterministic not just in the forward direction (as any other deterministic program), but also in the backward direction. A central consequence of this property is that well-formed programs must have both uniquely defined forward and backward semantics, with backward semantics given either directly or indirectly (*e.g.*, as is often done, by providing a textual translation of terms into terms which carry their inverse semantics; this approach is related to program inversion [23, 24]). In the case of reversible functional programming, reversibility is accomplished by guaranteeing local (forward and backward) determinism of evaluation – which, in turn, leads to global (forward and backward) determinism. Though reversible functions are injective [6], injectivity itself (a *global* property) is not enough to guarantee reversibility (a *local* property) – specifically, locally reversible control structures are necessary [22].

One such reversible functional programming language is RFUN, developed in recent years by Yokoyama, Axelsen, and Glück [22]. RFUN is an untyped language that uses Lisp-style symbols and constructors for data representation. Programs in RFUN are first-order functions, in which bound variables must be linearly used (though patterns are not required to be exhaustive). To account for the fact that data duplication *can* be performed reversibly, a *duplication-equality* operator [25], defined as follows, is used:

$$\lfloor \langle x \rangle \rfloor = \langle x, x \rangle$$
$$\lfloor \langle x, y \rangle \rfloor = \begin{cases} \langle x \rangle & \text{if } x = y \\ \langle x, y \rangle & \text{otherwise} \end{cases}$$

In the first case, the application of $\lfloor \cdot \rfloor$ to the unary tuple $\langle x \rangle$ yields the binary tuple $\langle x, x \rangle$, that is, the value x is duplicated. In the second case, when $x = y$, the application to $\langle x, y \rangle$ joins two identical values into $\langle x \rangle$; otherwise, the two values are returned unchanged (two different values cannot have been obtained by duplication of one value). Using an explicit operator simplifies reverse computation because the duplication of a value in one direction requires an equality check in the other direction, and vice versa. Instead of using a variable twice to duplicate a value, the duplication is made explicit. The operator is self-inverse, *e.g.*, $\lfloor \lfloor \langle x \rangle \rfloor \rfloor = \langle x \rangle$ and $\lfloor \lfloor \langle x, y \rangle \rfloor \rfloor = \langle x, y \rangle$.

The only control structure available in RFUN is a reversible case-expression employing the *symmetric first-match policy*: The control expression is matched against the patterns in the order they are given (as in,

```
plus ⟨x, y⟩ ≜ case y of
                Z    → ⌊⟨x⟩⌋
                S(u) → let ⟨x′, u′⟩ = plus ⟨x, u⟩ in
                       ⟨x′, S(u′)⟩

fib n ≜ case n of
          Z    → ⟨S(Z), S(Z)⟩
          S(m) → let ⟨x, y⟩ = fib m in
                 let z = plus ⟨y, x⟩ in z
```

Figure 1: The Fibonacci-pair program and its helper function $\mathit{plus}\ \langle x, y\rangle = \langle x, x + y\rangle$ in RFUN.

```
type Bool = False | True
type Nat  = 0 | Succ Nat

not :: Bool ↔ Bool
  | False ↔ True
  | True  ↔ False

parity :: Nat × Bool ↔ Nat × Bool
  | (n, b)                ↔ iter (n, 0, b)
  | iter (Succ n, m, b)   ↔ iter (n, Succ m, not b)
  | iter (0, m, b)        ↔ (m, b)
    where iter :: Nat × Nat × Bool ↔ Nat × Nat × Bool
```

Figure 2: The parity program in Theseus and its type definitions and helper function *not* :: Bool ↔ Bool.

e.g., the ML-family of languages), but, for the case-expression to be defined, once a match is found, any value produced by the matching branch must *not* match patterns that could have been produced by a previous branch. This policy guarantees reversibility. Perhaps surprising is the fact that recursion works in RFUN completely analogously to the way it works irreversibly; *i.e.*, using a call stack. In particular, inversion of recursive functions is handled simply by replacing the recursive call with a call to the inverse, and inverting the remainder of the function body. As such, the inverse of a recursive function is, again, a recursive function. This point will prove important later on.

An example of an RFUN program for computing Fibonacci-pairs is shown in Figure 1 [22, 25]: Given a natural number n encoded in unary, $\mathit{fib}(n)$ produces the pair $\langle f_{n+1}, f_{n+2}\rangle$ where f_i is the unary encoding of the i'th Fibonacci number. Notice the use of the duplication operator in the definition of *plus*: The duplication-equality operator on the right-hand side of the first branch of *plus* duplicates $\langle x\rangle$ into $\langle x, x\rangle$ in the forward direction, and checks the equality of two values $\langle x, y\rangle$ in the backward direction. This accounts for the fact that the first branch of *plus* always returns two identical values, while the second branch always returns two different values. The first-match policy of RFUN described above guarantees the reversibility of the auxiliary function *plus*, which is defined by $\mathit{plus}\ \langle x, y\rangle = \langle x, x + y\rangle$.

A different approach to reversible functional programming is given by Theseus, a language developed recently by James & Sabry [15] on top of the Π^0 reversible combinator calculus [16, 26]. Unlike RFUN, Theseus features a simple type system with products and sums as well as the unit and empty type, and supports user-defined (isorecursive) data types (declared in a style similar to Haskell and languages in the ML-family). Like RFUN, bound variables must be linearly used, but unlike RFUN, pattern clauses must be *exhaustive* and *non-overlapping*, properties that when combined makes both guaranteeing local reversibility and producing inverse programs straightforward.

Though Theseus is, like RFUN, a first-order language, an elegant feature with a flavor of higher-order programming is its support for *parametrized maps*, *i.e.*, functions that depend statically on data of a given type in order to produce a reversible function. For example, though ordinary function composition cannot be performed reversibly without production of garbage, if we know the functions to compose no later than at compile time, we can certainly produce their composition at compile time without additional garbage. This is handled in the Theseus type system by allowing both irreversible and (first order) reversible arrow types, with the proviso that all irreversible arrows be discharged at compile time. In this way, a parametrized function $\Omega :: (a \leftrightarrow b) \to (a \leftrightarrow b)$ is *not* a Theseus program, but if $f :: a \leftrightarrow b$ is a first-order reversible function, then so is $\Omega\ f :: a \leftrightarrow b$.

Recursion in Theseus is achieved using *typed iteration labels* (a feature unique to Theseus) that specify the type and behaviour of intermediate, tail-recursive computations. This feature is perhaps best illustrated by an example: Figure 2 shows a Theseus-program (courtesy of [15]) that recursively computes the parity of a natural number (encoded in unary). Though this approach sacrifices the possibility for expressing programs with nested recursion, the benefit is that all thus specified recursive reversible functions can be expressed as a non-recursive function that a †-trace operator is applied to (*cf.* [15, 16, 26] for details regarding this

operator specifically in the context of Theseus and Π^0). That is to say, if $f :: a \leftrightarrow b$ is a recursive Theseus program with an iteration label of type u, we may construct a (non-recursive) function $f' :: a + u \leftrightarrow b + u$ such that the trace of f' (tracing out u, the type of the iteration label) is precisely f. This eases the process of obtaining inverse semantics greatly, as we only need to take the trace of the inverse of the (non-recursive, so straightforward to invert) function f' to obtain the inverse of the recursive function f.

3. Background

This section gives an introduction to restriction and inverse categories (with joins), dagger categories, and categories of partial maps as they will be used in the remainder of this text. Unless otherwise stated, the material described in this section can be found in numerous texts on restriction and inverse category theory (*e.g.*, Cockett & Lack [17, 27, 28], Giles [18], Guo [19]).

3.1. Restriction and inverse categories

We begin by recalling the definition of *restriction structures* and *restriction categories*.

Definition 1 (Cockett & Lack). *A* restriction structure *on a category consists of an operator $\overline{(\cdot)}$ on morphisms mapping each morphism $f : A \to B$ to a morphism $\overline{f} : A \to A$ (the* restriction idempotent *of f) such that*

(i) $f \circ \overline{f} = f$ *for all morphisms* $f : A \to B$,

(ii) $\overline{f} \circ \overline{g} = \overline{g} \circ \overline{f}$ *whenever* $\mathrm{dom}(f) = \mathrm{dom}(g)$,

(iii) $\overline{f \circ \overline{g}} = \overline{f} \circ \overline{g}$ *whenever* $\mathrm{dom}(f) = \mathrm{dom}(g)$, *and*

(iv) $\overline{h} \circ f = f \circ \overline{h \circ f}$ *whenever* $\mathrm{cod}(f) = \mathrm{dom}(h)$.

A category with a restriction structure is called a restriction category.

As a trivial example, any category can be equipped with a restriction structure given by setting $\overline{f} = 1_A$ for every morphism $f : A \to B$. However, there are also many useful and nontrivial examples of restriction categories (see, *e.g.*, [17, Sec. 2.1.3]), the canonical one being the category **Pfn** of sets and partial functions. In this category, the restriction idempotent $\overline{f} : A \to A$ for a partial function $f : A \to B$ is given by the partial identity function

$$\overline{f}(x) = \begin{cases} x & \text{if } f \text{ is defined at } x, \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{1}$$

A related example is the category **PTop** of topological spaces and partial continuous functions (see also [18]). A partial function between topological spaces $f : X \to Y$ is continuous if

(i) the domain of definition of f (the set of points $X' \subseteq X$ for which f is defined) is open in X, and

(ii) f is continuous in the usual sense, *i.e.*, the set $f^{-1}(V) \subseteq X$ is open when $V \subseteq Y$ is.

Under this definition, **PTop** is a restriction category, with restriction idempotents given precisely as in Eq. (1); that a partial continuous function f is defined on an open set ensures precisely that $\overline{f}$ is continuous as well (and defined on an open set). Other examples include the category **DCPO** of pointed directed-complete partial orders and strict continuous maps, slice categories $\mathscr{C}/A$ for an object A of a restriction category $\mathscr{C}$ [17], and any inverse monoid (see also [29]) viewed as a (one-object) category.

Since we take restrictions as additional structure, we naturally want a notion of functors that preserve this structure.

Definition 2. *A functor $F : \mathscr{C} \to \mathscr{D}$ between restriction categories $\mathscr{C}$ and $\mathscr{D}$ is a* restriction functor *if $\overline{F(f)} = F(\overline{f})$ for all morphisms f of $\mathscr{C}$.*

A morphism $f : A \to B$ of a restriction category is said to be *total* if $\overline{f} = 1_A$. Given a restriction category $\mathscr{C}$, we can form the category Total($\mathscr{C}$), consisting of all objects and only the total morphisms of $\mathscr{C}$, which embeds in $\mathscr{C}$ via a faithful restriction functor. Restriction categories with restriction functors form a category, **rCat**.

Moving on to inverse categories, in order to define these[1] we first need the notion of a partial isomorphism:

Definition 3. *In a restriction category $\mathscr{C}$, we say that a morphism $f : A \to B$ is a* partial isomorphism *if there exists a unique morphism $f^\circ : B \to A$ of $\mathscr{C}$ (the* partial inverse *of f) such that $f^\circ \circ f = \overline{f}$ and $f \circ f^\circ = \overline{f^\circ}$.*

Definition 4. *A restriction category $\mathscr{C}$ is said to be an* inverse category *if all morphisms of $\mathscr{C}$ are partial isomorphisms.*

In this manner, if we accept an intuition of restriction categories as "categories with partiality," inverse categories are "groupoids with partiality" – and, indeed, the category **PInj** of sets and partial injective functions is *the* canonical example of an inverse category. In fact, the Wagner-Preston representation theorem (see, *e.g.*, [29]) for inverse monoids can be extended to show that every locally small inverse category can be faithfully embedded in **PInj** (see the two independent proofs by Kastl [30] and Heunen [31], or Cockett & Lack [17] for the special case of small inverse categories).

The analogy with groupoids goes even further; similar to how we can construct a groupoid Core($\mathscr{C}$) by taking only the isomorphisms of $\mathscr{C}$, every restriction category $\mathscr{C}$ has a subcategory Inv($\mathscr{C}$) that is an inverse category with the same objects as $\mathscr{C}$, and all partial isomorphisms of $\mathscr{C}$ as morphisms.

More generally, inverse categories are *dagger categories* (sometimes also called *categories with involution*):

Definition 5. *A category $\mathscr{C}$ is said to be a* dagger category *if it is equipped with a contravariant endofunctor $(-)^\dagger : \mathscr{C}^{\mathrm{op}} \to \mathscr{C}$ such that $1_A^\dagger = 1_A$ and $f^{\dagger\dagger} = f$ for all morphisms f and identities 1_A.*

Note that this definition in particular implies that a dagger functor must act as the identity on objects[2]

Proposition 1. *Every inverse category $\mathscr{C}$ is a dagger category with the dagger functor given by $A^\dagger = A$ on objects, and $f^\dagger = f^\circ$ on morphisms.*

As is conventional, we will call $f^\dagger$ the *adjoint* of f, and say that f is *self-adjoint* (or *hermitian*) if $f = f^\dagger$, and *unitary* if $f^\dagger = f^{-1}$. In inverse categories, unitary morphisms thus correspond precisely to (total) isomorphisms. For the remainder of this text, we will use this induced dagger-structure when refering to the partial inverse of a morphism (and write, *e.g.*, $f^\dagger$ rather than f°).

A useful feature of this definition of inverse categories is that we do not need an additional notion of an "inverse functor" as a functor that preserves partial inverses; restriction functors suffice.

Proposition 2. *Every restriction functor $F : \mathscr{C} \to \mathscr{D}$ between inverse categories preserves the canonical dagger structure of $\mathscr{C}$,* i.e., *$F(f)^\dagger = F(f^\dagger)$ for all morphisms f of $\mathscr{C}$.*

A simple argument for this proposition is the fact that partial isomorphisms are defined purely in terms of composition and restriction idempotents, both of which are preserved by restriction functors. Inverse categories with restriction functors form a category, **invCat**.

[1] Strictly speaking, inverse categories predate restriction categories – see Kastl [30] for the first published article on inverse categories to the knowledge of the authors. Though we will use the axiomatization following from restriction categories, inverse categories can equivalently be defined as the categorical extension of inverse monoids, *i.e.*, as categories where all morphisms have a regular inverse, and all idempotents commute.

[2] Though it may look as if we are superfluously demanding preservation of identities at first glance, what we are stating is something stronger, namely that the dagger functor must map identities in $\mathscr{C}^{\mathrm{op}}$ to *themselves* in $\mathscr{C}$. That is, we are requiring $1_A^\dagger = 1_A$ rather than $1_A^\dagger = 1_{A^\dagger}$.

3.2. Split restriction categories and categories of partial maps

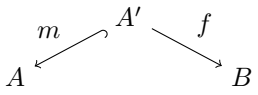
When working in a category with a distinguished class of idempotents, it is often desirable that they split[3]. In restriction categories, such a distinguished class is given by the class of restriction idempotents, leading us to the straightforward definition of a split restriction category:

Definition 6. *A restriction category in which every restriction idempotent splits is called a* split restriction category*.*

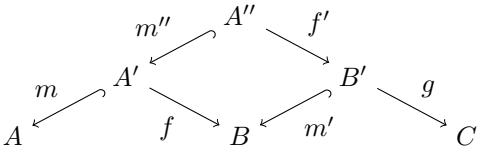
For example, when equipped with the usual restriction structure, **Pfn** and **PTop** are both split restriction categories, though it straightforward to come up with subcategories of either that are not. It follows, by way of the Karoubi envelope, that every restriction category $\mathscr{C}$ can be embedded in a split restriction category $\mathrm{Split}(\mathscr{C})$ via a fully faithful restriction functor (though this requires us to show that $\mathrm{Split}(\mathscr{C})$ inherits the restriction structure from $\mathscr{C}$, and that the fully faithful functor into this category preserves restrictions; see Prop. 2.26 of [17] for details). Prime examples of split restriction categories are categories of partial maps.

Categories of partial maps provide a synthetic approach to partiality in a categorical setting [32]. To form a category of partial maps, we consider a *stable system of monics*: In a category $\mathscr{C}$, a collection $\mathcal{M}$ of monics of $\mathscr{C}$ is said to be a stable system of monics if it contains all isomorphisms of $\mathscr{C}$ and is closed under composition and pullbacks (in the sense that the pullback m' of an $m : X \to B$ in $\mathcal{M}$ along any $f : A \to B$ exists and $m' \in \mathcal{M}$). Given such a stable system of monics $\mathcal{M}$ in a category $\mathscr{C}$, we can form the category of partial maps as follows:

Proposition 3. *Given a category $\mathscr{C}$ and a stable system of monics $\mathcal{M}$ of $\mathscr{C}$, we form the category of partial maps $\mathrm{Par}(\mathscr{C}, \mathcal{M})$ by choosing the objects to be the objects of $\mathscr{C}$, and placing a morphism $(m, f) : A \to B$ for every pair (m, f) where $m : A' \to A \in \mathcal{M}$ and $f : A' \to B$ is a morphism of $\mathscr{C}$, as in*



factored out by the equivalence relation $\cdot \sim \cdot$ in which $(m, f) \sim (m', f')$ if there exists an isomorphism $\alpha : A' \to A''$ such that $m' \circ \alpha = m$ and $f' \circ \alpha = f$. Composition of morphisms $(m, f) : A \to B$ and $(m', g) : B \to C$ is given by $(m \circ m'', g \circ f') : A \to C$ where m'' and f' arise from the pullback



where $m'' \circ m \in \mathcal{M}$ precisely by $\mathcal{M}$ closed under composition and pullbacks.

Categories of partial maps are prime examples of restriction categories; in fact, of split restriction categories. Even further, every split restriction category is isomorphic to a category of partial maps [17]. As previously noted, every restriction category can be fully and faithfully embedded in a split restriction category, and consequently, in a category of partial maps.

[3]Recall that a splitting of an idempotent $e : A \to A$ consists of an object A' and morphisms $m : A \to A'$ and $r : A' \to A$ such that $r \circ m = e$ and $m \circ r = \mathrm{id}_{A'}$.

3.3. Partial order enrichment, joins, and compatibility

A useful feature of restriction categories, and one we will exploit throughout this article, is that hom-sets can be equipped with a partial order, defined as follows:

Proposition 4. *In a restriction category $\mathscr{C}$, every hom-set $\mathrm{Hom}_{\mathscr{C}}(A,B)$ can be equipped with the structure of a partial order where we let $f \leq g$ iff $g \circ \overline{f} = f$. Further, every restriction functor F is locally monotone with respect to this order, in the sense that $f \leq g$ implies $F(f) \leq F(g)$.*

In **Pfn**, this corresponds to the usual partial order on partial functions: For $f, g : A \to B$, $f \leq g$ if, for all $x \in A$, f is defined at x implies that g is defined at x and $f(x) = g(x)$.

A natural question to ask is when this partial order has a least element: A sufficient condition for this is when the restriction category has a *restriction zero*.

Definition 7. *A restriction category $\mathscr{C}$ has a* restriction zero *object* 0 *iff for all objects A and B, there exists a* unique *morphism $0_{A,B} : A \to B$ that factors through 0 and satisfies $\overline{0_{A,A}} = 0_{A,A}$.*

If such a restriction zero object exists, it is unique up to (total) isomorphism (as it is a zero object in the usual sense). When a given restriction category has such a restriction zero, the zero map $0_{A,B} : A \to B$ is precisely the least element of $\mathrm{Hom}_{\mathscr{C}}(A,B)$. Note that it may seem more natural to require instead that $\overline{0_{A,B}} = 0_{A,A}$ for all A and B. This is equivalent to requiring $\overline{0_{A,A}} = 0_{A,A}$:

Lemma 1. *When a restriction category has the zero object, $\overline{0_{A,A}} = 0_{A,A}$ if and only if $\overline{0_{A,B}} = 0_{A,A}$.*

Proof. Supposing $\overline{0_{A,B}} = 0_{A,A}$ for all A and B, this directly implies $\overline{0_{A,A}} = 0_{A,A}$ for all A. In the other direction, we observe by the universal mapping property of the zero object that $0_{A,B} = 0_{A,B} \circ 0_{A,A}$, so

$$\overline{0_{A,B}} = \overline{0_{A,B} \circ 0_{A,A}} = \overline{0_{A,B} \circ \overline{0_{A,A}}} = \overline{0_{A,B}} \circ \overline{0_{A,A}} = \overline{0_{A,B}} \circ 0_{A,A} = 0_{A,A}$$

by the assumption of $\overline{0_{A,A}} = 0_{A,A}$ and the universal mapping property of the zero object. □

Given that hom-sets of restriction (and, by extension, inverse) categories are partially ordered, one may wonder when this partial order has joins. It turns out, however, that it does not in the general case, and that only very simple restriction categories have joins for arbitrary parallel morphisms. However, we *can* define a meaningful notion of joins for parallel morphisms if this operation is not required to be total, but only be defined for *compatible* morphisms. Nevertheless, these partial joins turn out to be tremendously useful, and will prove key in many of the constructions in the following sections. For restriction categories, this compatibility relation is defined as follows:

Definition 8. *Parallel morphisms $f, g : A \to B$ of a restriction category $\mathscr{C}$ are said to be* restriction compatible *if $g \circ \overline{f} = f \circ \overline{g}$; if this is the case, we write $f \smile g$. By extension, a set $S \subseteq \mathrm{Hom}_{\mathscr{C}}(A,B)$ is restriction compatible, or $\smile$-compatible, if all morphisms of S are pairwise restriction compatible.*

This compatibility relation can be extended to apply to inverse categories by requiring that morphisms be compatible in both directions:

Definition 9. *Parallel morphisms $f, g : A \to B$ of an inverse category $\mathscr{C}$ are said to be* inverse compatible *if $f \smile g$ and $f^\dagger \smile g^\dagger$; if this is the case, we write $f \asymp g$. A set $S \subseteq \mathrm{Hom}_{\mathscr{C}}(A,B)$ is inverse compatible, or $\asymp$-compatible, if all morphisms of S are pairwise inverse compatible.*

The familiar reader will notice that this definition differs in its statement from Guo's [19, p. 98], who defined $f \asymp g$ in an inverse category $\mathscr{C}$ if $f \smile g$ holds in both $\mathscr{C}$ and $\mathscr{C}^{\mathrm{op}}$ (relying on the observation that inverse categories are simultaneously restriction categories and *co*restriction categories). To avoid working explicitly with corestriction categories, however, we will use this equivalent definition instead.

We define (countable) restriction joins and (countable) join restriction categories as follows:

Definition 10 (Guo). *Say that a restriction category $\mathscr{C}$ is outfitted with (countable)* $\sim$-joins *if for all (countable) $\sim$-compatible subsets S of all hom sets $\mathrm{Hom}_{\mathscr{C}}(A,B)$ (for a compatibility relation $\cdot \sim \cdot$), there exists a morphism $\bigvee_{s\in S} s$ such that*

(i) $s \leq \bigvee_{s\in S} s$ *for all* $s \in S$*, and* $s \leq t$ *for all* $s \in S$ *implies* $\bigvee_{s\in S} s \leq t$*;*

(ii) $\overline{\bigvee_{s\in S} s} = \bigvee_{s\in S} \overline{s}$*;*

(iii) $f \circ \left(\bigvee_{s\in S} s\right) = \bigvee_{s\in S}(f \circ s)$ *for all* $f : B \to X$*; and*

(iv) $\left(\bigvee_{s\in S} s\right) \circ g = \bigvee_{s\in S}(s \circ g)$ *for all* $g : Y \to A$*.*

Definition 11. *A restriction category is said to be a (countable)* join restriction category *if it has (countable) $\smile$-joins.*

In addition, we say that a restriction functor that preserves all thus constructed joins is a *join restriction functor*. Notice that the join of the empty set (which is vacuously compatible for any compatibility relation), which we will tellingly denote $0_{A,B} : A \to B$, is always the least element with respect to the partial order on hom-sets. When a join restriction category has a restriction zero object, empty joins and zero maps coincide.

As a concrete example, **Pfn** has joins of all restriction compatible sets; here, $f \smile g$ iff whenever f and g are both defined at some point x, $f(x) = g(x)$, and the join of a set of restriction compatible partial functions F is given by

$$\left(\bigvee_{f\in F} f\right)(x) = \begin{cases} f'(x) & \text{if there exists an } f' \in F \text{ such that } f' \text{ is defined at } x, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Notice that the compatibility relation ensures precisely that the result is a partial function. This construction extends to the category **PTop** of topological spaces and partial continuous functions defined on open sets: That the resulting function is defined on an open set follows by the fact that arbitrary unions are open, and that it is continous follows by openness and the *gluing* (or *pasting*) *lemma*. As such, **PTop** has all joins as well.

This, finally, allows us to define join inverse categories by narrowing the definition above to only require the existence of joins of inverse compatible (sets of) morphisms:

Definition 12. *An inverse category is said to be a (countable)* join inverse category *if has (countable) $\asymp$-joins.*

Analogously to **Pfn**, the category **PInj** is a join inverse category with joins given precisely as in **Pfn**, since the additional requirement that $f^\dagger \smile g^\dagger$ ensures that the resulting partial function is injective.

4. As DCPO-categories

In the present section, we will show that inverse categories with countable joins are intrinsically **DCPO**-enriched. This observation leads to two properties that are highly useful for modelling reversible functional programming, namely the existence of fixed points for both morphism schemes for recursion (that is, continuous endomorphisms on hom-objects) and for locally continuous functors. The former can be applied to model reversible recursive functions, and the latter to model recursive data types [33]. Further, we will show that the partial inverse of the fixed point of a morphism scheme for recursion can be computed as the fixed point of an *adjoint* morphism scheme, giving a style of recursion similar to RFUN as discussed in Section 2.

Recall that a category is **DCPO**-enriched (or simply a **DCPO**-category) if all hom-sets are pointed directed complete partial orders (*i.e.*, they have a least element and satisfy that each directed subset has a supremum), and composition is continuous and strict (recall that continuous functions are monotone by

definition). For full generality, though the countable case will suffice for our applications[4], for some cardinal κ we let $\mathbf{DCPO}_\kappa$ denote the category of pointed directed κ-complete partial orders (*i.e.*, partially ordered sets with a least element satisfying that every directed subset of cardinality at most κ has a supremum) with strict and continuous maps. To begin, we will need the lemma below.

Lemma 2. *In any inverse category, partial inversion is monotone with respect to the natural partial order in the sense that $f \leq g$ implies $f^\dagger \leq g^\dagger$.*

Proof. Suppose $f \leq g$, *i.e.*, $g \circ \overline{f} = f$ by definition. Then

$$\begin{aligned} g^\dagger \circ \overline{f^\dagger} = g^\dagger \circ f \circ f^\dagger = g^\dagger \circ g \circ \overline{f} \circ f^\dagger = \overline{g} \circ \overline{f} \circ f^\dagger = \overline{g \circ \overline{f}} \circ f^\dagger \\ = \overline{f} \circ f^\dagger = f^\dagger \circ f \circ f^\dagger = f^\dagger \circ \overline{f^\dagger} = f^\dagger \end{aligned}$$

so $f^\dagger \leq g^\dagger$ as well, as desired. □

We also recall some basic properties of least elements of hom-sets in join inverse categories:

Lemma 3. *Let $\mathscr{C}$ be an inverse (or restriction) category with (at least empty) joins, and let $0_{A,B}$ be the least element (*i.e.*, the empty join) of $\mathrm{Hom}_{\mathscr{C}}(A,B)$. Then*

(i) $f \circ 0_{A,B} = 0_{A,Y}$ *for all* $f : B \to Y$,

(ii) $0_{A,B} \circ g = 0_{X,B}$ *for all* $g : X \to A$, *and*

(iii) $0_{A,B}$ *has a partial inverse* $0_{A,B}^\dagger = 0_{B,A}$.

Proof. For (i), since $0_{A,B}$ is the empty join, it follows that $f \circ 0_{A,B} = f \circ \bigvee_{s \in \emptyset} s = \bigvee_{s \in \emptyset} (f \circ s) = 0_{A,Y}$, and completely analogously for (ii). For (iii), let $0_{B,A}$ be the least element in $\mathrm{Hom}_{\mathscr{C}}(A,B)$. Then

$$0_{B,A} \circ 0_{A,B} = \left(\bigvee_{s \in \emptyset} s\right) \circ \left(\bigvee_{t \in \emptyset} t\right) = \left(\bigvee_{s \in \emptyset} s\right) \circ 0_{A,B} = \bigvee_{s \in \emptyset} (s \circ 0_{A,B}) = 0_{A,A} = \bigvee_{t \in \emptyset} \overline{t} = \overline{\bigvee_{t \in \emptyset} t} = \overline{0_{A,B}},$$

and by entirely analogous argument, $0_{A,B} \circ 0_{B,A} = 0_{B,B} = \overline{0_{B,A}}$. □

These lemmas allow us to show $\mathbf{DCPO}_\kappa$-enrichment (for some cardinal κ) of inverse categories with joins of directed sets of parallel morphisms with cardinality at most κ:

Theorem 4. *Every inverse (or restriction) category satisfies that if it has joins of compatible sets of cardinality at most κ respectively all joins of compatible sets, it is enriched in $\mathbf{DCPO}_\kappa$ respectively $\mathbf{DCPO}$.*

Proof. Let A, B be objects of $\mathscr{C}$, and let $F \subseteq \mathrm{Hom}_{\mathscr{C}}(A,B)$ be directed (with respect to the canonical partial ordering) – of cardinality at most κ, if required. Let $f, g : A \to B$ be in F. Since F is directed, there exists an $h : A \to B$ in F such that $f \leq h$ and $g \leq h$, *i.e.*, $h \circ \overline{f} = f$ and $h \circ \overline{g} = g$. But then $g \circ \overline{f} = h \circ \overline{g} \circ \overline{f} = h \circ \overline{f} \circ \overline{g} = f \circ \overline{g}$ so $f \smile g$; by Lemma 2 we have $f^\dagger \leq h^\dagger$ and $g^\dagger \leq h^\dagger$ as well, so $f^\dagger \smile g^\dagger$ follows entirely analogously. Thus $f \asymp g$, so F is $\asymp$-compatible, allowing us to form the supremum of F by

$$\sup F = \bigvee_{f \in F} f$$

which is the supremum of this directed set directly by definition of the join.

[4]Strictly speaking, enrichment in ω-CPOs is sufficient to show all results that follow from Theorem 4, *i.e.*, Corollary 5, Theorem 7, and Theorem 14. Notably, only countable joins (rather than arbitrary joins) are needed to obtain these results.

Monotony of compositions holds in all restriction categories, not just inverse categories with countable joins: Supposing $f \leq g$ then $g \circ \overline{f} = f$, and for $h : B \to X$,

$$h \circ g \circ \overline{h \circ f} = h \circ g \circ \overline{h \circ g \circ \overline{f}} = h \circ g \circ \overline{h \circ g} \circ \overline{f} = h \circ g \circ \overline{f} = h \circ f$$

so $h \circ f \leq h \circ g$ in $\mathrm{Hom}_{\mathscr{C}}(A, X)$; the argument is analogous for postcomposition. That composition is continuous follows by monotony and definition of joins, as we have

$$h \circ \sup \{f\}_{f \in F} = h \circ \bigvee_{f \in F} f = \bigvee_{f \in F} (h \circ f) = \sup \{h \circ f\}_{f \in F}$$

for all $h : B \to X$, and analogously for postcomposition. That composition is strict follows by Lemma 3.

Recall that a functor $F : \mathscr{C} \to \mathscr{D}$ between **DCPO**-categories is *locally continuous* iff each $F_{A,B} : \mathrm{Hom}_{\mathscr{C}}(A, B) \to \mathrm{Hom}_{\mathscr{D}}(FA, FB)$ is continuous (so specifically, all locally continuous functors are locally monotone). Note that since all restriction functors preserve the partial order on hom-sets, and since suprema are defined in terms of joins, join restriction functors are in particular locally continuous.

4.1. Reversible fixed points of morphism schemes

In the following, let $\mathscr{C}$ be an inverse category with countable joins – so, by Theorem 4, enriched in $\mathbf{DCPO}_{\aleph_0}$. We will use the term *morphism scheme* to refer to a continuous function $f : \mathrm{Hom}_{\mathscr{C}}(A, B) \to \mathrm{Hom}_{\mathscr{C}}(X, Y)$ – note that such schemes are morphisms of $\mathbf{DCPO}_{\aleph_0}$ and not of the inverse category $\mathscr{C}$, so they are specifically *not* required to have inverses. Enrichment in $\mathbf{DCPO}_{\aleph_0}$ then has the following immediate corollary by Kleene's fixed point theorem:

Corollary 5. *Every morphism scheme of the form $f : \mathrm{Hom}_{\mathscr{C}}(A, B) \to \mathrm{Hom}_{\mathscr{C}}(A, B)$ has a least fixed point $\mathrm{fix}\, f : A \to B$ in $\mathscr{C}$.*

Proof. Define $\mathrm{fix}\, f = \sup \{f^n(0_{A,B})\}_{n \in \omega}$; this is an ω-chain, so specifically a directed set of cardinality at most $\aleph_0$. That this is the least fixed point follows by Kleene's fixed point theorem, as $0_{A,B}$ is the least element in $\mathrm{Hom}_{\mathscr{C}}(A, B)$. □

Morphism schemes on their own are useful for modelling parametrized reversible functions, as discussed in relation to Theseus in Section 2. Under this interpretation, recursive reversible functions can be seen as the least fixed points of self-parametrized reversible functions.

Given that we can thus model reversible recursive functions via least fixed points of morphism schemes, a prudent question to ask is if the inverse of a least fixed point can be computed as the least fixed point of another morphism scheme. We will answer this in the affirmative, but to do this, we need to show that the induced dagger functor is locally continuous.

Lemma 6. *The canonical dagger functor $\dagger : \mathscr{C}^{\mathrm{op}} \to \mathscr{C}$ is locally continuous.*

Proof. Let $F \subseteq \mathrm{Hom}_{\mathscr{C}}(A, B)$ be directed with respect to the canonical partial order. As local monotony was already shown in Lemma 2, it suffices to show that suprema are preserved. Since $f \leq \bigvee_{f \in F} f$ for each $f \in F$ by Definition 12, we have $f^\dagger \leq \left(\bigvee_{f \in F} f\right)^\dagger$ for all $f \in F$ by monotony of $\dagger$, and so

$$\sup \{f^\dagger\}_{f \in F} = \bigvee_{f \in F} f^\dagger \quad \leq \quad \left(\bigvee_{f \in F} f\right)^\dagger = \sup \{f\}^\dagger_{f \in F}$$

by Definition 12. In the other direction, we have $f^\dagger \leq \bigvee_{f \in F} f^\dagger$ for all $f \in F$ by Definition 12, so by monotony of $\dagger$, $f = f^{\dagger\dagger} \leq \left(\bigvee_{f \in F} f^\dagger\right)^\dagger$ for all $f \in F$. But then $\bigvee_{f \in F} f \leq \left(\bigvee_{f \in F} f^\dagger\right)^\dagger$ by Definition 12, and so by

monotony of †, we finally get

$$\sup\,\{f\}^{\dagger}_{f\in F} = \left(\bigvee_{f\in F} f\right)^{\dagger} \quad \leq \quad \left(\bigvee_{f\in F} f^{\dagger}\right)^{\dagger\dagger} = \bigvee_{f\in F} f^{\dagger} = \sup\,\{f^{\dagger}\}_{f\in F}$$

as desired. □

With this lemma, we are able to show that the inverse of a least fixed point of a morphism scheme can be computed as the least fixed point of an adjoint morphism scheme:

Theorem 7. *Every morphism scheme of the form* $f : \mathrm{Hom}_{\mathscr{C}}(A,B) \to \mathrm{Hom}_{\mathscr{C}}(A,B)$ *has an adjoint morphism scheme* $f_{\ddagger} : \mathrm{Hom}_{\mathscr{C}}(B,A) \to \mathrm{Hom}_{\mathscr{C}}(B,A)$ *such that* $(\mathrm{fix}\, f)^{\dagger} = \mathrm{fix}\, f_{\ddagger}$.

Proof. Let $\iota_{A,B} : \mathrm{Hom}_{\mathscr{C}}(A,B) \to \mathrm{Hom}_{\mathscr{C}}(B,A)$ denote the family of functions defined by $\iota_{A,B}(f) = f^{\dagger}$; each of these are continuous by Lemma 6, and an isomorphism (with inverse $\iota_{B,A}$) by $f^{\dagger\dagger} = f$. Given a morphism scheme $f : \mathrm{Hom}_{\mathscr{C}}(A,B) \to \mathrm{Hom}_{\mathscr{C}}(A,B)$, we define $f_{\ddagger} = \iota_{A,B} \circ f \circ \iota_{B,A}$ – this is continuous since it is a (continuous) composition of continuous functions. But since

$$f^{n}_{\ddagger} = (\iota_{A,B} \circ f \circ \iota_{B,A})^{n} = \iota_{A,B} \circ f^{n} \circ \iota_{B,A}$$

by $\iota_{B,A}$ an isomorphism with inverse $\iota_{A,B}$, and since $0^{\dagger}_{A,B} = 0_{B,A}$ by Lemma 3, we get

$$\begin{aligned}\mathrm{fix}\, f_{\ddagger} &= \sup\,\{f^{n}_{\ddagger}(0_{B,A})\}_{n\in\omega} = \sup\,\{(\iota_{A,B} \circ f^{n} \circ \iota_{B,A})(0_{B,A})\}_{n\in\omega} = \sup\,\{f^{n}(0^{\dagger}_{B,A})^{\dagger}\}_{n\in\omega} \\ &= \sup\,\{f^{n}(0_{A,B})^{\dagger}\}_{n\in\omega} = \sup\,\{f^{n}(0_{A,B})\}^{\dagger}_{n\in\omega} = (\mathrm{fix}\, f)^{\dagger}\end{aligned}$$

as desired. □

In modelling recursion in reversible functional programming, this theorem states precisely that the partial inverse of a recursive reversible function is, itself, a recursive reversible function, and that it can be obtained by inverting the function body and replacing recursive calls with recursive calls to the thus constructed inverse: This is *precisely* the inverse semantics of recursive reversible functions in RFUN, as discussed in Section 2.

4.2. Algebraic ω-compactness for free!

A pleasant property of **DCPO**-categories is that algebraic ω-compactness – the property that every locally continuous functor has a canonical fixed point – is relatively easy to check, thanks to the fixed point theorem due to Adámek [33] and Barr [34]:

Theorem 8 (Adámek & Barr). *Let* $\mathscr{C}$ *be a* (**D**)**CPO***-category with an initial object. If* $\mathscr{C}$ *has colimits of* ω*-sequences of embeddings, then* $\mathscr{C}$ *is algebraically* ω*-compact over* (**D**)**CPO**.

Note that this theorem was originally stated for **CPO**-categories; categories in which every hom-set is an ω-CPO (in the sense that every ω-chain of parallel morphisms has a supremum), and composition is continuous and strict. However, since ω-chains are but specific examples of directed sets, every **DCPO**-category is a **CPO**-category, and every functor that is locally continuous with respect to **DCPO**-enrichment is locally continuous with respect to **CPO**-enrichment as well. By the same argument, noting that ω-chains are directed sets of cardinality at most $\aleph_0$, it suffices to be $\mathbf{DCPO}_{\aleph_0}$-enriched.

Recall that an *embedding* in a (**D**)**CPO**-category is a morphism $e : A \to B$ with a *projection* $p : B \to A$ such that $p \circ e = 1_A$ and $e \circ p \sqsubseteq 1_B$ (as such, with the canonical (**D**)**CPO**-enrichment, embeddings in join restriction categories are specifically total – in fact, they correspond to *restriction monics* in the sense of [17]).

Canonical fixed points of functors are of particular interest in modelling functional programming, since they can be used to provide interpretations for recursive data types. In the following, we will couple this theorem with a join-completion theorem for restriction categories to show that every inverse category can be faithfully embedded in an algebraically ω-compact inverse category with joins. That this succeeds rests on the following lemmas:

Lemma 9. *There is an adjunction*

$$\mathbf{rCat} \underset{U}{\overset{\mathrm{Inv}}{\underset{\top}{\rightleftarrows}}} \mathbf{invCat}$$

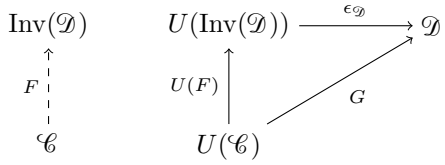
between the forgetful functor $U : \mathbf{invCat} \to \mathbf{rCat}$ *and the functor* $\mathrm{Inv} : \mathbf{rCat} \to \mathbf{invCat}$ *that maps a restriction category to its subcategory of partial isomorphisms.*

Proof. Let $F : U(\mathscr{C}) \to \mathscr{D}$ be a restriction functor between some inverse category $\mathscr{C}$ and restriction category $\mathscr{D}$. As $\mathscr{C}$ is an inverse category, the restriction category $U(\mathscr{C})$ contains only partial isomorphisms, which F has to preserve (as it is a restriction functor); as such, the image of F in $\mathscr{D}$ lies in the inverse subcategory $\mathrm{Inv}(\mathscr{D})$, so we may simply define the functor $\mathscr{C} \to \mathrm{Inv}(\mathscr{D})$ to act precisely as F on both objects and morphisms.

In the other direction, given $G : \mathscr{C} \to \mathrm{Inv}(\mathscr{D})$, we define the restriction functor $U(\mathscr{C}) \to \mathscr{D}$ by letting it act as G on both objects and morphisms; nothing further is required, as inverse categories are extensionally restriction categories.

That is determines a natural isomorphism follows immediately by the fact that neither direction actually alters how the given functor acts on objects or morphisms. □

An immediate consequence of this adjunction is that the inverse subcategory $\mathrm{Inv}(\mathscr{D})$ of $\mathscr{D}$ is uniquely determined (up to canonical isomorphism) as the largest inverse subcategory of $\mathscr{D}$, in the sense that for any other inverse category $\mathscr{C}$ with a restriction functor $G : U(\mathscr{C}) \to \mathscr{D}$, there is a unique functor $F : \mathscr{C} \to \mathrm{Inv}(\mathscr{D})$ such that the following diagram commutes



where the counit $\epsilon_{\mathscr{C}}$ is the obvious faithful inclusion functor. For our purposes, it gives the following corollary:

Corollary 10. *If an inverse category* $\mathscr{C}$ *embeds faithfully in a restriction category* $\mathscr{D}$*, it also embeds faithfully in* $\mathrm{Inv}(\mathscr{D})$*.*

Proof. By Lemma 9, it suffices to show that $F : \mathscr{C} \to \mathrm{Inv}(\mathscr{D})$ is faithful when $G : U(\mathscr{C}) \to \mathscr{D}$ is. Suppose G is faithful; by the universal mapping property, $G = \epsilon_{\mathscr{C}} \circ U(F)$ for a unique $F : \mathscr{C} \to \mathrm{Inv}(\mathscr{D})$. Since G is faithful by assumption, $U(F)$ must be faithful as well, in turn implying that F is faithful (as U is the forgetful functor).

Lemma 11. *The functor* $\mathrm{Inv} : \mathbf{rCat} \to \mathbf{invCat}$ *takes join restriction categories to join inverse categories (and preserves join restriction functors).*

Proof. It suffices to show that if S is a set of inverse compatible parallel partial isomorphisms, then $\bigvee_{s \in S} s$ is a partial isomorphism – this follows directly by continuity of partial inversion (see Lemma 6). □

The latter of these lemmas was also shown by Guo [19, Lemma 3.1.27]. As for the completion theorem, in order to ease presentation we make the following notational shorthand.

Convention 13. *For a restriction category* $\mathscr{C}$*, let* $\overline{\mathscr{C}}$ *denote the category of presheaves* $\mathbf{Set}^{\mathrm{Total}(\mathrm{Split}(\mathscr{C}))^{\mathrm{op}}}$*.*

Note that $\overline{\mathscr{C}}$ is cocomplete and all colimits are stable under pullback (since colimits in $\overline{\mathscr{C}}$ are constructed object-wise in **Set**). This is used in the completion theorem for join restriction categories, due to Cockett and Guo [19, 35].

Theorem 12 (Cockett & Guo). *Every restriction category $\mathscr{C}$ can be faithfully embedded in a join restriction category of the form* $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{gap}})$.

The stable system of monics $\widehat{\mathcal{M}_{\mathrm{gap}}}$ relates to the join-completion for restriction categories via $\mathcal{M}$-gaps (see Cockett [35] or Guo [19, Sec. 3.2.2] for details).

Lemma 13. *For a a split restriction category* $\mathrm{Par}(\mathscr{C}, \mathcal{M})$*, the following are equivalent:*

(i) Every hom-set of $\mathrm{Par}(\mathscr{C}, \mathcal{M})$ *has a least element,*

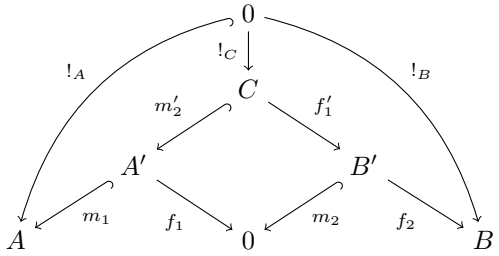
(ii) $\mathscr{C}$ *has a strict initial object* 0 *and each morphism* $!_A : 0 \to A$ *is in* $\mathcal{M}$*, and*

(iii) $\mathrm{Par}(\mathscr{C}, \mathcal{M})$ *has a restriction zero object.*

Proof. (i) $\Leftrightarrow$ (ii): See Guo [19], Lemmas 3.3.1 and 3.3.2.

(ii) $\Rightarrow$ (iii): We will show that 0 is a restriction zero in $\mathrm{Par}(\mathscr{C}, \mathcal{M})$ with the zero map $0_{A,B} : A \to B$ given by the span $A \xleftarrow{!_A} 0 \xrightarrow{!_B} B$. Suppose that some morphism $(m, f) : A \to B$ factors through 0 in $\mathrm{Par}(\mathscr{C}, \mathcal{M})$ as $(m_2, f_2) \circ (m_1, f_1)$, *i.e.*, we are in a situation indicated by the bottom part of the diagram below in $\mathscr{C}$, where $(m_1 \circ m_2', f_2 \circ f_1') \sim (m, f)$ (*i.e.*, there exists an isomorphism α in $\mathscr{C}$ such that $m_1 \circ m_2' \circ \alpha = m$ and $f_2 \circ f_1' \circ \alpha = f$).

But since $f_1 \circ m_2' = m_2 \circ f_1' : C \to 0$ is a morphism into the strict initial object 0, it must be an isomorphism, with only possible inverse the unique map $1_C : 0 \to C$. Since $!_A = !_C \circ m_2' \circ m_1$ and $!_B = !_C \circ f_1' \circ f_2$ the universal mapping property of the initial object, it follows that the isomorphism $!_C$ witnesses $(!_A, !_B) \sim (m_1 \circ m_2', f_2 \circ f_1') \sim (m, f)$, so 0 is the zero object in $\mathrm{Par}(\mathscr{C}, \mathcal{M})$. That it is the restriction zero follows by the fact that the restriction structure on $\mathrm{Par}(\mathscr{C}, \mathcal{M})$ is given by $\overline{(m, f)} = (m, m)$, so $0_{A,A} = (!_A, !_A) = \overline{0_{A,A}}$.

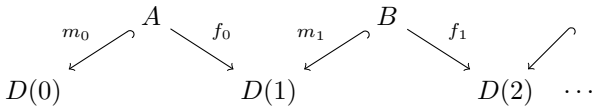


(iii) $\Rightarrow$ (i): Let $0_{A,B} : A \to B$ be the unique zero map for arbitrarily chosen objects A and B, and let $f : A \to B$ be any other morphism. By Lemma 1, $\overline{0_{A,B}} = 0_{A,A}$ for a restriction zero, so $f \circ \overline{0_{A,B}} = f \circ 0_{A,A} = 0_{A,B}$ by the universal mapping property of the zero object; thus $0_{A,B} \leq f$, and hence $0_{A,B}$ is the least element in $\mathrm{Hom}(A, B)$. □

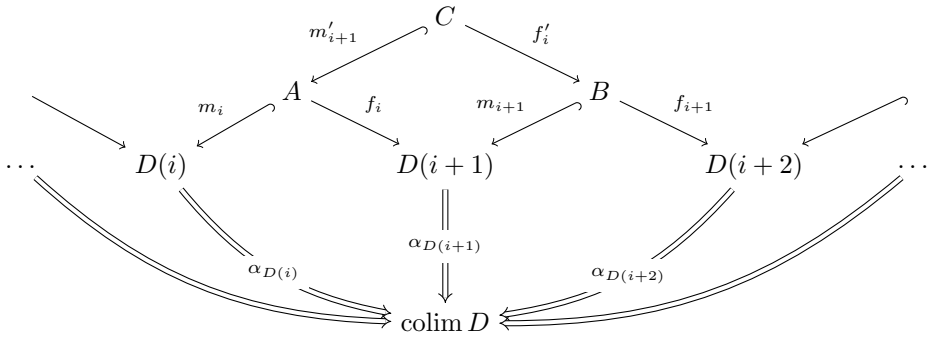
We can now show the algebraic ω-compactness theorem for restriction categories:

Theorem 14. *If* $\mathscr{C}$ *is a restriction category then* $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{gap}})$ *is algebraically* ω*-compact over* **DCPO***, and thus over join restriction functors.*

Proof. Let $\mathscr{C}$ be a restriction category. By Theorem 12, $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}})$ is a join restriction category, and since it has all joins (specifically all empty joins), it follows that it has a restriction zero object 0 (which is specifically initial) by Lemma 13. By Adámek & Barr's fixed point theorem and the fact that restriction categories with arbitrary joins are **DCPO**-enriched (by Theorem 4), it suffices to show that $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}})$ has colimits of ω-diagrams of embeddings. Let $D : \omega \to \mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}})$ be such a diagram of embeddings. This corresponds to the diagram



in $\overline{\mathscr{C}}$. Since $\overline{\mathscr{C}}$ is cocomplete, this diagram has a colimiting cocone $\alpha : D \Rightarrow \operatorname{colim} D$ such that the diagram below commutes. Further, since colimits in $\overline{\mathscr{C}}$ are constructed object-wise in **Set**, this colimit is stable under pullbacks, so the pullbacks in $\overline{\mathscr{C}}$ used for composition in $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}})$ commutes with this colimit. Since embeddings are total, each m_i is an isomorphism, and since isomorphisms are stable under pullback, any m'_i arising from a pullback (correponding to composition) is an isomorphism as well. As such, any $m_i \circ m'_{i+1}$ is an isomorphism, and so the isomorphism $m'^{(-1)}_{i+1}$ witnesses $(m_i \circ m'_{i+1}, \alpha_{D(i+2)} \circ f_{i+1} \circ f'_i) \sim (m_i, \alpha_{D(i+1)} \circ f_i)$; iterating this argument for arbitrary finite k, we see that everything commutes, as desired. Thus, the family of morphisms $\{(m_i, \alpha_{D(i+1)} \circ f_i)\}_{i \in \omega}$ is a colimiting cocone for D in $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}})$.



□

Finally, using this machinery, we can show how this theorem extends to inverse categories.

Corollary 15. *Every inverse category can be faithfully embedded in a join inverse category that is algebraically ω-compact over join restriction functors.*

Proof. Let $\mathscr{C}$ be an inverse category. Since $U(\mathscr{C})$ is the exact same category viewed as a restriction category, $U(\mathscr{C})$ embeds faithfully in $\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}})$, which is a join restriction category by Theorem 12, and algebraically ω-compact by Theorem 14. But then it follows by Corollary 10 that $\mathscr{C}$ embeds faithfully in $\mathrm{Inv}(\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}}))$, which is a join inverse category by Lemma 11, and is algebraically ω-compact for join restriction functors (which are specifically locally continuous) since fixed points of functors are (total) isomorphisms, so preserved in $\mathrm{Inv}(\mathrm{Par}(\overline{\mathscr{C}}, \widehat{\mathcal{M}_{\mathrm{gap}}}))$. □

4.3. Applications in models of reversible functional programming

This final corollary shows directly that join inverse categories are consistent with algebraic ω-compactness over join restriction functors, which can thus be used to model recursive data types in the style of Theseus as well as RFUN terms, given the existence of suitable join-preserving monoidal functors as follows: Suppose we are given an inverse product $(\otimes, 1)$ and a disjointness tensor $(\oplus, 0)$, both join-preserving (see [18] for details on both; the definition of latter can also found in Definition 17 in the following section) such that there are (total) natural isomorphisms

$$A \otimes (B \oplus C) \overset{\delta_{A,B,C}}{\cong} (A \otimes B) \oplus (A \otimes C) \qquad\qquad A \otimes 0 \overset{\nu_A}{\cong} 0$$

giving the category the structure of a *bimonoidal category* (or *rig category*; see also [36, 37]). Using the compactness theorem from before, this enables us to model all (isorecursive) data types expressible in Theseus

by modelling product types using the inverse product, sum types using the disjointness tensor, the unit type as 1, the empty type as 0, and recursive types using the canonical fixed point. For example, the two types from the Theseus example in Figure 2, defined as

$$\begin{aligned}\textbf{type } \underline{\text{Bool}} &= \text{False} \mid \text{True}\\ \textbf{type } \underline{\text{Nat}} &= 0 \mid \text{Succ } \underline{\text{Nat}}\end{aligned}$$

can both be given very familiar denotations as $1 \oplus 1$ respectively $\mu X.1 \oplus X$ (*i.e.*, the least – in this case, unique – fixed point of the functor defined by $F(X) = 1 \oplus X$).

Though RFUN is untyped, with terms formed using Lisp-style symbols and constructors, this bimonoidal structure is also helpful in constructing a denotation of the universal type of RFUN terms. Formally, RFUN terms are constructed inductively as follows: Given a denumerable set $\mathbb{S}$ of symbols, a term t is either a symbol (*i.e.*, $t \in \mathbb{S}$) or of the form $c(t_1, t_2, \ldots, t_k)$ for some finite k, where $c \in \mathbb{S}$ and each t_i is a term. Supposing that $\mathbb{S}$ is the Kleene star closure of the latin alphabet, examples of terms include a, $s(s(z))$, and $f(o, o(bar, baz))$.

Since the symbols $\mathbb{S}$ is denumerable, we can identify symbols one-to-one with natural numbers – and so with the object $\mu X.1 \oplus X$ using algebraic compactness. Further, in the familiar way, we can define the functor mapping an object A to lists of A by $A \mapsto \mu X.1 \oplus (A \otimes X)$. This, finally, means that we can define a functor mapping an object A to terms over A by $A \mapsto \mu X.A \otimes L(X)$.

Moreover, we saw how this view of join restriction categories as **DCPO**-categories also enabled the style of reversible recursion found in RFUN, exemplified in the Fibonacci-pair program in Figure 1. More specifically, to give the denotation of a recursive function in RFUN, instead of trying to interpret it directly as an endomorphism on the term object $T \to T$, we instead interpret it as a morphism scheme $\text{Hom}(T,T) \xrightarrow{[\![f']\!]} \text{Hom}(T,T)$ in which the recursive call is replaced by a call to the morphism given as argument, and then take the least fixed point of this morphism scheme fix $[\![f']\!]$ as the denotation of the recursive function f. Further details on this construction, and the other constructions sketched in this section, will appear in a forthcoming paper.

5. As unique decomposition categories

Complementary to the view on inverse categories with countable joins as **DCPO**-categories, we will show that these can also be viewed as unique decomposition categories, a kind of category introduced by Haghverdi [20] equipped with a partial sum operation on hom-sets via enrichment in the category of Σ-monoids (shown to be symmetric monoidal by Hoshino [21]). Unique decomposition categories (including Hoshino's *strong* unique decomposition categories [21] which we will employ here) are specifically traced monoidal categories [38] if they satisfy certain conditions. This is desirable when modelling functional programming, as traces can be used to model notions of feedback [39] and recursion [40–42].

Here, we will show that inverse categories with countable joins and a join-preserving *disjointness tensor* (due to Giles [18]) are strong unique decomposition categories, and satisfy the conditions required to be equipped with a trace. We extend this result further to show that the trace is a †-trace [43], and thus has pleasant inversion properties (the trace in **PInj** is well studied, *cf.* [20, 44, 45]). This is particularly interesting given that the reversible programming language Theseus [15] and the combinator calculus Π^0 [16] both rely on a †-trace for reversible recursion.

We begin with the definition of a Σ-monoid [20] (see also Manes & Benson [46] where these first appeared as *positive partial monoids*):

Definition 14. *A* Σ-monoid (M, Σ) *consists of a nonempty set* M *and a partial operator* Σ *defined on countable families in* M *(say that a family* $\{x_i\}_{i\in I}$ *is* summable *if* $\sum_{i\in I} x_i$ *is defined) such that*

(i) if $\{x_i\}_{i\in I}$ *is a countable family in* M *and* $\{I_j\}_{j\in J}$ *is a countable partitioning of* I*, then* $\{x_i\}_{i\in I}$ *is summable iff all* $\{x_i\}_{i\in I_j}$ *and* $\sum_{i\in I_j} x_i$ *are summable for all* $j \in J$*, and in this case*

$$\sum_{j\in J}\sum_{i\in I_j} x_i = \sum_{i\in I} x_i\,,$$

(ii) any family $\{x_i\}_{i\in I}$ *in* M *where* I *is singleton is summable with* $\sum_{i\in I} x_i = x_j$ *if* $I = \{j\}$.

The class of Σ-monoids with homomorphisms preserving partial sums forms a category, Σ**Mon**. As such, a category $\mathscr{C}$ is enriched in Σ**Mon** if all hom-sets of $\mathscr{C}$ are Σ-monoids, and composition distributes over partial addition.

Lemma 16. *Every inverse category with countable joins is* Σ**Mon***-enriched.*

Proof. Let $\mathscr{C}$ be an inverse category with countable joins. Let $\{s_i\}_{i\in I}$ be a countable family of morphisms taken from $\mathrm{Hom}_{\mathscr{C}}(A, B)$ for some objects A, B of $\mathscr{C}$ and countable index set I. We define

$$\sum_{i\in I} s_i = \bigvee_{s\in\{s_i|i\in I\}} s$$

so, by definition, summability coincides with join compatibility.

To see that axiom (i) of Definition 14 is satisfied, let $\{I_j\}_{j\in J}$ be a partitioning of I and suppose that $\{s_i\}_{i\in I}$ is summable, *i.e.*, inverse compatible. By definition of inverse compatibility, this means that all morphisms of $\{s_i\}_{i\in I}$ are pairwise inverse compatible, and since all partition families $\{s_i\}_{i\in I_j}$ for $j \in J$ consist only of morphisms taken from $\{s_i\}_{i\in I}$, they are summable by all $\{s_i \mid i \in I_j\}$ inverse compatible; that

$$\sum_{j\in J}\sum_{i\in I_j} x_i = \sum_{i\in I} x_i$$

follows by the least upper bound property of the join (Definition 12 (i)).

Conversely, suppose that all $\{s_i\}_{i\in I_j}$ and all $\sum_{i\in I_j} s_i$ are summable for all $j \in J$. Let f and g be arbitrary morphisms of $\{s_i\}_{i\in I}$; then, f is an element of a partition $\{s_i\}_{i\in I_j}$ for $j \in J$, and g is an element of a partition $\{s_i\}_{i\in I_k}$ for $k \in J$. If $j = k$ then f and g are inverse compatible by $\{s_i\}_{i\in I_j}$ summable – if $j \neq k$, we have $f \leq \bigvee_{s\in\{s_i|i\in I_j\}} s = \sum_{i\in I_j} s_i$ so f and $\bigvee_{s\in\{s_i|i\in I_j\}} s$ are inverse compatible by Lemma 2, and g and $\bigvee_{s\in\{s_i|i\in I_k\}} s$ are inverse compatible by an analogous argument. But then f and g are inverse compatible by $\bigvee_{s\in\{s_i|i\in I_j\}} s = \sum_{i\in I_j} s_i$ and $\bigvee_{s\in\{s_i|i\in I_k\}} s = \sum_{i\in I_k} s_i$ summable (*i.e.*, inverse compatible) by assumption, and by transitivity of join compatibility. The summation identity follows, once again, using Definition 12 (i).

For axiom (ii) of Definition 14, this follows by $f \leq f$ for any morphism $f : A \to B$, and so for a singleton $F = \{f\}$ (and such a singleton always exists, since every hom-set has a least element given by the empty join), $f \leq \bigvee_{s\in F} s$ and $\bigvee_{s\in F} s \leq f$ both follow by Definition 12 (i), so $f = \bigvee_{s\in F} s$. That composition distributes over partial addition follows directly by Definition 12 (iii) and (iv). □

Haghverdi defines unique decomposition categories in the following way:

Definition 15 (Haghverdi). *A unique decomposition category* $\mathscr{C}$ *is a symmetric monoidal category enriched in* Σ**Mon** *such that for all finite index sets* I *and all* $j \in I$*, there are* quasi-injections $\iota_j : X_j \to \oplus_{i\in I} X_i$ *and* quasi-projections $\rho_j : \oplus_{i\in I} X_i \to X_j$ *satisfying*

(i) $\rho_k \circ \iota_j = 1_{X_k}$ *if* $j = k$*, and* $0_{X_j,X_k}$ *otherwise, and*

(ii) $\Sigma_{i\in I}\iota_i \circ \rho_i = 1_{\oplus_{i\in I} X_i}$.

By slight abuse of notation, we will use $0_{A,B} : A \to B$ to denote the morphism arising from summing the empty family of $\mathrm{Hom}_{\mathscr{C}}(A, B)$. That the empty family is always summable – and that its sum serves as unit – follows from axioms (i) *and* (ii) of Definition 14, as (ii) and the assumption of nonemptiness guarantees the summability of at least one family, while (i) ensures that any partitioning of this family – including into empty partitions – is summable when the family is (so the empty family is summable), and that the sum of summed partitions coincides with the sum of the original family (giving us that the sum of the empty family is the unit; see also [46] or [20]). This allows us to state the strengthened definition by Hoshino:

Definition 16 (Hoshino). *A strong unique decomposition category is a symmetric monoidal category enriched in* Σ**Mon** *satisfying that the identity on the monoidal unit* I *is* $0_{I,I}$*, and that*

$$1_X \oplus 0_{Y,Y} + 0_{X,X} \oplus 1_Y = 1_{X \oplus Y}$$

for all X *and* Y*.*

An elementary result is that strong unique decomposition categories are unique decomposition categories, with their quasi injections and quasi projections given by $\iota_1 = (1_A \oplus 0_{0,B}) \circ u_r^{-1} : A \to A \oplus B$ and $\rho_1 = u_r \circ (1_A \oplus 0_{B,0}) : A \oplus B \to A$ (where $u_r : A \oplus 0 \to A$ is the right unitor of the monoidal functor $- \oplus -$), and analogously for ι_2 and ρ_2 (thus extending to any finite index set).

As such, (strong) unique decomposition categories rely on a sum-like monoidal tensor – in the context of inverse categories, such a one can be found in Giles' definition of a disjointness tensor [18, Definition 7.2.1].

Definition 17 (Giles). *An inverse category* $\mathscr{C}$ *with a restriction zero object* 0 *is said to have a* disjointness tensor *if it is equipped with a symmetric monoidal restriction functor* $- \oplus - : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$ *such that*

(i) the restriction zero 0 *is the tensor unit, and*

(ii) the morphisms $\amalg_1 : A \to A \oplus B$ *and* $\amalg_2 : A \to B \oplus A$ *given by* $\amalg_1 = (1_A \oplus 0_{0,B}) \circ u_r^{-1}$ *and* $\amalg_2 = (0_{0,B} \oplus 1_A) \circ u_l^{-1}$ *are jointly epic, and their partial inverses* $\amalg_1^\dagger : A \oplus B \to A$ *and* $\amalg_2^\dagger : B \oplus A \to A$ *are jointly monic,*

where $u_l : 0 \oplus A \to A$ *and* $u_r : A \oplus 0 \to A$ *denote the left respectively the right unitor of the symmetric monoidal tensor.*

Though not required from this definition, since we are working exclusively with join inverse categories, we make the additional assumption that the disjointness tensor is a join restriction functor. Since Giles' definition already demands that the zero object be the monoidal unit, and even defines $\amalg_i$ precisely like Hoshino's definition of ι_i (one can similarly see that $\amalg_i^\dagger = \rho_i$), we can show the following:

Theorem 17. *Every inverse category with countable joins and a join-preserving disjointness tensor is a strong unique decomposition category.*

Proof. By Lemma 16, any inverse category with countable joins (and a join-preserving disjointness tensor) is enriched in Σ**Mon**, so it suffices to show that the (specifically symmetric monoidal) disjointness tensor satisfies Definition 16. That $1_{I,I} = 0_{I,I}$ follows by $1_{0,0} = 0_{0,0}$ for the (restriction) zero 0, and $1_X \oplus 0_{Y,Y} + 0_{X,X} \oplus 1_Y = 1_{X \oplus Y}$ by the definition of partial sums as joins and the additional requirement that the disjointness tensor preserves joins. □

Due to the Σ**Mon**-enrichment on unique decomposition categories, the trace can be constructed as a denumerable sum of morphisms, provided that morphisms of a certain form are always summable, *cf.* [20, Prop. 4.0.11] and [21, Corr. 5.4]:

Theorem 18 (Haghverdi, Hoshino). *Let* $\mathscr{C}$ *be a (strong) unique decomposition category such that for every* X*,* Y*, and* U *and every* $f : X \oplus U \to Y \oplus U$*, the sum* $f_{11} + \sum_{n=0}^{\infty} f_{21} \circ f_{22}^n \circ f_{12}$ *exists, where* $f_{ij} = \rho_j \circ f \circ \iota_i$*. Then* $\mathscr{C}$ *has a uniform trace given by*

$$\mathrm{Tr}^U_{X,Y}(f) = f_{11} + \sum_{n=0}^{\infty} f_{21} \circ f_{22}^n \circ f_{12}\,.$$

In a restriction category, we say that parallel morphisms $f, g : A \to B$ are *disjoint* if $\overline{f} \circ \overline{g} = 0_{A,A}$.

Lemma 19. *In a restriction category, the following hold:*

(i) All disjoint morphisms are restriction compatible,

(ii) $g \smile g'$ *and* $f \smile f'$ *implies* $g \circ f \smile g' \circ f'$ *when* $\mathrm{dom}(g) = \mathrm{cod}(f)$*, and*

(iii) $\overline{g \circ f} = \overline{g \circ f} \circ \overline{f}$ *when* $\mathrm{dom}(g) = \mathrm{cod}(f)$*.*

Proof. For (i), suppose $f, g : A \to B$ are disjoint, *i.e.*, $\overline{f} \circ \overline{g} = 0_{A,A}$. Then

$$g \circ \overline{f} = g \circ \overline{g} \circ \overline{f} = g \circ \overline{f} \circ \overline{g} = g \circ 0_{A,A} = 0_{A,B} = f \circ 0_{A,A} = f \circ \overline{f} \circ \overline{g} = f \circ \overline{g}.$$

Part (ii) was shown by Guo [19, Lemma 3.1.3]. For (iii) we have $\overline{g \circ f} = \overline{g \circ f \circ \overline{f}} = \overline{g \circ f} \circ \overline{f}$, as desired. □

This lemma allows us to show that all join inverse categories are traced monoidal categories with a uniform trace.

Theorem 20. *Every inverse category with countable joins and a disjointness tensor has a uniform trace.*

Proof. By Theorem 18, it suffices to show that all morphisms of the forms f_{11} or $f_{21} \circ f_{22}^n \circ f_{12}$ for any $n \in \mathbb{N}$ and some $f : X \oplus U \to Y \oplus U$ are pairwise inverse compatible. We notice that

$$(f_{ij})^\dagger = (\rho_j \circ f \circ \iota_i)^\dagger = (\amalg_j^\dagger \circ f \circ \amalg_i)^\dagger = \amalg_i^\dagger \circ f^\dagger \circ \amalg_j^{\dagger\dagger} = \amalg_i^\dagger \circ f^\dagger \circ \amalg_j = (f^\dagger)_{ji}$$

and so $(f_{11})^\dagger = (f^\dagger)_{11}$ and

$$(f_{21} \circ f_{22}^n \circ f_{12})^\dagger = (f_{12})^\dagger \circ (f_{22}^n)^\dagger \circ (f_{21})^\dagger = (f^\dagger)_{21} \circ (f_{22}^\dagger)^n \circ (f^\dagger)_{12}$$

so it suffices to show only restriction compatibility, since the restriction compatibility of the partial inverses will follow directly by this symmetry.

To see that $f_{11} \smile f_{21} \circ f_{22}^k \circ f_{12}$ for some $k \in \mathbb{N}$, notice that $f_{11} = \amalg_1^\dagger \circ f \circ \amalg_1$ and

$$f_{21} \circ f_{22}^k \circ f_{12} = f_{21} \circ f_{22}^k \circ \amalg_2^\dagger \circ f \circ \amalg_1$$

so it suffices by Lemma 19 to show that $\amalg_1^\dagger \smile f_{21} \circ f_{22}^n \circ \amalg_2^\dagger$. But then

$$\overline{f_{21} \circ f_{22}^n \circ \amalg_2^\dagger} \circ \overline{\amalg_1^\dagger} = \overline{f_{21} \circ f_{22}^n \circ \amalg_2^\dagger} \circ \overline{\amalg_2^\dagger} \circ \overline{\amalg_1^\dagger} = 0_{Y \oplus U, Y \oplus U}$$

since $\overline{\amalg_1^\dagger} = \amalg_1 \circ \amalg_1^\dagger = 1_Y \oplus 0_{U,U}$ and $\overline{\amalg_2^\dagger} = \amalg_2 \circ \amalg_2^\dagger = 0_{Y,Y} \oplus 1_U$, so these are restriction compatible by Lemma 19.

To see that $f_{21} \circ f_{22}^m \circ f_{12} \smile f_{21} \circ f_{22}^n \circ f_{12}$, assume without loss of generality that $m < n$ (the case where $m = n$ is trivial). Once again, by Lemma 19, it suffices to show $f_{21} \smile f_{21} \circ f_{22}^{n-m}$. But since

$$f_{21} = \amalg_1^\dagger \circ f \circ \amalg_2$$

and

$$f_{21} \circ f_{22}^{n-m} = f_{21} \circ (\amalg_2^\dagger \circ f \circ \amalg_2)^{(n-m)-1} \circ \amalg_2^\dagger \circ f \circ \amalg_2$$

restriction compatibility follows by analogous argument to the previous case. □

Recall that a †-category with a trace is said to have a †-*trace* (see, *e.g.*, [43]) if $\mathrm{Tr}_{X,Y}^U(f)^\dagger = \mathrm{Tr}_{Y,X}^U(f^\dagger)$ for every morphism $f : X \oplus U \to Y \oplus U$.

Theorem 21. *The canonical trace in an inverse category with countable joins and a disjointness tensor is a* †*-trace.*

PROOF. To see that the trace induced by Theorems 18 and 20 is a †-trace, let $f : X \oplus U \to Y \oplus U$ be a morphism of $\mathscr{C}$.

In the proof of Theorem 20, we noticed that $(f_{ij})^\dagger = (f^\dagger)_{ji}$ and $(f_{21} \circ f_{22}^n \circ f_{12})^\dagger = (f^\dagger)_{21} \circ (f_{22}^\dagger)^n \circ (f^\dagger)_{12}$. Expanding this in the definition of the canonical trace given by Theorem 18, we get

$$\begin{aligned}
\mathrm{Tr}^U_{X,Y}(f)^\dagger &= \left(f_{11} + \sum_{n\in\omega} f_{21} \circ f_{22}^n \circ f_{12}\right)^\dagger = \left(f_{11} \vee \bigvee_{n\in\omega} f_{21} \circ f_{22}^n \circ f_{12}\right)^\dagger \\
&= (f_{11})^\dagger \vee \left(\bigvee_{n\in\omega} f_{21} \circ f_{22}^n \circ f_{12}\right)^\dagger = (f_{11})^\dagger \vee \bigvee_{n\in\omega} (f_{12})^\dagger \circ (f_{22}^n)^\dagger \circ (f_{21})^\dagger \\
&= (f^\dagger)_{11} \vee \bigvee_{n\in\omega} (f^\dagger)_{21} \circ (f^\dagger)_{22}^n \circ (f^\dagger)_{12} = \mathrm{Tr}^U_{Y,X}(f^\dagger)
\end{aligned}$$

by definition of the partial sum as join (Lemma 16), and by $(\bigvee_{f\in F} f)^\dagger = \bigvee_{f\in F} f^\dagger$ by Lemma 6. □

5.1. Applications in models of reversible functional programming

This final theorem is highly relevant to modelling Theseus in join inverse categories, as the *iteration label*-approach to reversible tail recursion (exemplified in the parity program in Figure 2) is equivalent to the existence of a †-trace operator. This can be observed from the fact that we are not only able to provide a forward and backward semantics to functions with iteration labels via a †-trace [15] (see also the discussion in Section 2), but that it is also possible to express a †-trace operator as a parametrized function (which, in turn, can be naturally regarded categorically as a morphism scheme) in Theseus [15].

To give a concrete example, consider the recursive parity function in Theseus from Figure 2. To give semantics to its recursive behaviour using a †-trace, we systematically transform from a function of type $\underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}} \leftrightarrow \underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}}$ with an internal iteration label of type $\underline{\mathrm{Nat}} \times \underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}} \leftrightarrow \underline{\mathrm{Nat}} \times \underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}}$ into a function of type $(\underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}}) + (\underline{\mathrm{Nat}} \times \underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}}) \leftrightarrow (\underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}}) + (\underline{\mathrm{Nat}} \times \underline{\mathrm{Nat}} \times \underline{\mathrm{Bool}})$ by prefacing patterns for the outer (parity) function by **Left**, replacing patterns for the inner (iteration label) function by **Right**-patterns, replacing calls to the inner function by **Right**-expressions, and prefacing return values by **Left**-expressions, as in the following:

```
parity :: Nat × Bool ↔ Nat × Bool
| (n, b)                  ↔ iter (n, 0, b)
| iter (Succ n, m, b)     ↔ iter (n, Succ m, not b)
| iter (0, m, b)          ↔ (m, b)
  where iter :: Nat × Nat × Bool ↔
                Nat × Nat × Bool
```

⇒

```
parity′ :: (Nat × Bool) + (Nat × Nat × Bool) ↔
           (Nat × Bool) + (Nat × Nat × Bool)
| Left (n, b)             ↔ Right (n, 0, b)
| Right (Succ n, m, b)    ↔ Right (n, Succ m, not b)
| Right (0, m, b)         ↔ Left (m, b)
```

Notice that this transformation preserves non-overlapping and exhaustive patterns, as are required of Theseus functions. In this way, we can obtain the denotation of the original *parity* function by taking the †-trace of the denotation of the transformed *parity′* function.

6. Conclusion

We have shown that inverse categories with countable joins carry with them a few key properties that are highly useful for modelling partial reversible functional programming. Notably, we have shown that any inverse category with countable joins is **DCPO**-enriched – from this view, we gathered that morphism schemes have fixed points, and that the partial inverses of such fixed points can be computed as fixed points of adjoint morphism schemes. This gave us a model of recursion à la RFUN.

Further, we were able to show that any inverse category can be embedded in an inverse category with joins, in which all join restriction functors have canonical fixed points. Finally, we showed that the presence of a join-preserving disjointness tensor on an inverse category with countable joins gives us a strong unique decomposition category, and in turn, a uniform †-trace: a model of recursion à la Theseus and Π^0.

Restriction categories have recently been considered as enriched categories by Cockett & Garner [47], though their approach relied on enrichments based on *weak double categories* rather than monoidal categories, as it is otherwise usually done (including in this paper). Further, fixed points in categories with a notion of partiality have previously been considered, notably by Fiore [48] who also relied on order-enrichment, though his work was in categories of partial maps directly. Finally, Giles [18] has shown the construction of a trace in inverse categories recently, relying instead on the presence of countable *disjoint sums* rather than joins (whether or not this approach leads to a †-trace is unspecified). It should also be noted that the trace in the canonical inverse category **PInj** has been studied independently of unique decomposition and restriction categories, notably by Hines [44] and Abramsky, Haghverdi, and Scott [45].

As regards future work, since an inverse category with countable joins and a disjointness tensor is †-traced, it can be embedded in a †-compact closed category via the Int-construction [38, 49]. It may be of interest to consider †-compact closed categories generated in this manner, as we suspect these will be inverse categories as well (notably, Int(**PInj**) is [44]) – and could provide, *e.g.*, an alternative treatment of projectors as restriction idempotents, and isometries as restriction monics (see also [50]).

Additionally, while our focus in this article has been on inverse categories, we conjecture that many of these results can be generalized to restriction categories.

Acknowledgments. The authors wish to thank the anonymous reviewers for their thoughtful and detailed comments, and the anonymous reviewers of FoSSaCS 2016 as well as the participants of NWPT 2015 for their useful comments on earlier versions of the paper. The research was partly funded by the *Danish Council for Independent Research* under the *Foundations of Reversible Computing* project. We also acknowledge the support given by *COST Action IC1405 Reversible Computation: Extending Horizons of Computing.*

References

[1] R. Kaarsgaard, Join inverse categories and reversible recursion, abstract presented at the 27th Nordic Workshop on Programming Theory (2015).
[2] H. B. Axelsen, R. Kaarsgaard, Join inverse categories as models of reversible recursion, in: B. Jacobs, C. Löding (Eds.), Foundations of Software Science and Computation Structures, 19th International Conference, FOSSACS 2016, Proceedings, Vol. 9634 of Lecture Notes in Computer Science, Springer, 2016, pp. 73–90.
[3] R. Landauer, Irreversibility and heat generation in the computing process, IBM Journal of Research and Development 5 (3) (1961) 183–191.
[4] E. Fredkin, T. Toffoli, Conservative logic, International Journal of Theoretical Physics 21 (3-4) (1982) 219–253.
[5] C. H. Bennett, Logical reversibility of computation, IBM Journal of Research and Development 17 (6) (1973) 525–532.
[6] H. B. Axelsen, R. Glück, What do reversible programs compute?, in: M. Hofmann (Ed.), Foundations of Software Science and Computational Structures, 14th International Conference, FOSSACS 2011, Proceedings, Vol. 6604 of Lecture Notes in Computer Science, Springer, 2011, pp. 42–56.
[7] M. Kutrib, M. Wendlandt, Reversible limited automata, in: J. Durand-Lose, B. Nagy (Eds.), Machines, Computations, and Universality, 7th International Conference, MCU 2015, Proceedings, Vol. 9288 of Lecture Notes in Computer Science, Springer, 2015, pp. 113–128.
[8] K. Morita, Two-way reversible multihead automata, Fundamenta Informaticae 110 (1–4) (2011) 241–254.
[9] M. Schordan, D. Jefferson, P. Barnes, T. Oppelstrup, D. Quinlan, Reverse code generation for parallel discrete event simulation, in: J. Krivine, J.-B. Stefani (Eds.), Reversible Computation, 7th International Conference, RC 2015, Proceedings, Vol. 9138 of Lecture Notes in Computer Science, Springer, 2015, pp. 95–110.
[10] I. Cristescu, J. Krivine, D. Varacca, A compositional semantics for the reversible π-calculus, in: LICS 2013, IEEE Computer Society, 2013, pp. 388–397.
[11] U. P. Schultz, M. Bordignon, K. Støy, Robust and reversible execution of self-reconfiguration sequences, Robotica 29 (1) (2011) 35–57.
[12] U. P. Schultz, J. S. Laursen, L. Ellekilde, H. B. Axelsen, Towards a domain-specific language for reversible assembly sequences, in: J. Krivine, J.-B. Stefani (Eds.), Reversible Computation, 7th International Conference, RC 2015, Proceedings, Vol. 9138 of Lecture Notes in Computer Science, Springer, 2015, pp. 111–126.
[13] T. Yokoyama, R. Glück, A reversible programming language and its invertible self-interpreter, in: Partial Evaluation and Program Manipulation. Proceedings, ACM, 2007, pp. 144–153.
[14] T. Yokoyama, H. B. Axelsen, R. Glück, Fundamentals of reversible flowchart languages, Theoretical Computer Science 611 (2016) 87–115.
[15] R. P. James, A. Sabry, Theseus: A high level language for reversible computing, work-in-progress report at RC 2014, available at `https://www.cs.indiana.edu/~sabry/papers/theseus.pdf`. (2014).
[16] W. J. Bowman, R. P. James, A. Sabry, Dagger traced symmetric monoidal categories and reversible programming, in: A. De Vos, R. Wille (Eds.), Reversible Computation 2011, Proceedings, Ghent University, 2011, pp. 51–56.

[17] J. R. B. Cockett, S. Lack, Restriction categories I: Categories of partial maps, Theoretical Computer Science 270 (1–2) (2002) 223–259.
[18] B. G. Giles, An investigation of some theoretical aspects of reversible computing, Ph.D. thesis, University of Calgary (2014).
[19] X. Guo, Products, joins, meets, and ranges in restriction categories, Ph.D. thesis, University of Calgary (2012).
[20] E. Haghverdi, A categorical approach to linear logic, geometry of proofs and full completeness, Ph.D. thesis, Carlton University and University of Ottawa (2000).
[21] N. Hoshino, A representation theorem for unique decomposition categories, in: U. Berger, M. Mislove (Eds.), MFPS XXVIII, Vol. 286 of Electronic Notes in Theoretical Computer Science, Elsevier, 2012, pp. 213–227.
[22] T. Yokoyama, H. B. Axelsen, R. Glück, Towards a reversible functional language, in: A. De Vos, R. Wille (Eds.), Reversible Computation, Third International Workshop, RC 2011, Revised papers, Vol. 7165 of Lecture Notes in Computer Science, Springer, 2012, pp. 14–29.
[23] S. M. Abramov, R. Glück, Principles of inverse computation and the universal resolving algorithm, in: T. Æ. Mogensen, D. A. Schmidt, I. H. Sudborough (Eds.), The Essence of Computation: Complexity, Analysis, Transformation, Vol. 2566 of Lecture Notes in Computer Science, Springer, 2002, pp. 269–295.
[24] R. Glück, M. Kawabe, Derivation of deterministic inverse programs based on LR parsing, in: Y. Kameyama, P. J. Stuckey (Eds.), Functional and Logic Programming, 7th International Symposium, FLOPS 2004, Proceedings, Vol. 2998 of Lecture Notes in Computer Science, Springer, 2004, pp. 291–306.
[25] R. Glück, M. Kawabe, A program inverter for a functional language with equality and constructors, in: A. Ohori (Ed.), Programming Languages and Systems, First Asian Symposium, APLAS 2003, Proceedings, Vol. 2895 of Lecture Notes in Computer Science, Springer, 2003, pp. 246–264.
[26] R. P. James, A. Sabry, Information effects, in: Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '12, ACM, 2012, pp. 73–84.
[27] J. R. B. Cockett, S. Lack, Restriction categories II: Partial map classification, Theoretical Computer Science 294 (1–2) (2003) 61–102.
[28] J. R. B. Cockett, S. Lack, Restriction categories III: Colimits, partial limits and extensivity, Mathematical Structures in Computer Science 17 (4) (2007) 775–817.
[29] M. V. Lawson, Inverse Semigroups: The Theory of Partial Symmetries, World Scientific, 1998.
[30] J. Kastl, Inverse categories, in: H.-J. Hoehnke (Ed.), Algebraische Modelle, Kategorien und Gruppoide, Vol. 7 of Studien zur Algebra und ihre Anwendungen, Akademie-Verlag, 1979, pp. 51–60.
[31] C. Heunen, On the functor ℓ^2, in: Computation, Logic, Games, and Quantum Foundations - The Many Facets of Samson Abramsky, Vol. 7860 of Lecture Notes in Computer Science, Springer, 2013, pp. 107–121.
[32] E. Robinson, G. Rosolini, Categories of partial maps, Information and Computation 79 (1988) 95–130.
[33] J. Adámek, Recursive data types in algebraically ω-complete categories, Information and Computation 118 (1995) 181–190.
[34] M. Barr, Algebraically compact functors, Journal of Pure and Applied Algebra 82 (3) (1992) 211–231.
[35] J. R. B. Cockett, X. Guo, Join restriction categories and the importance of being adhesive, presentation at Category Theory 2007 (2007).
[36] M. L. Laplaza, Coherence for distributivity, in: G. M. Kelly, M. L. Laplaza, G. Lewis, S. Mac Lane (Eds.), Coherence in Categories, Vol. 281 of Lecture Notes in Mathematics, Springer, 1972, pp. 29–65.
[37] J. Carette, A. Sabry, Computing with semirings and weak rig groupoids, in: P. Thiemann (Ed.), Programming Languages and Systems, 25th European Symposium on Programming, Proceedings, Vol. 9632 of Lecture Notes in Computer Science, Springer, 2016, pp. 123–148.
[38] A. Joyal, R. Street, D. Verity, Traced monoidal categories, Mathematical Proceedings of the Cambridge Philosophical Society 119 (3) (1996) 447–468.
[39] S. Abramsky, Retracing some paths in process algebra, in: U. Montanari, V. Sassone (Eds.), CONCUR '96: Concurrency Theory, 7th International Conference, Proceedings, Vol. 1119 of Lecture Notes in Computer Science, Springer, 1996, pp. 1–17.
[40] M. Hasegawa, Recursion from cyclic sharing: Traced monoidal categories and models of cyclic lambda calculi, in: P. de Groote, J. R. Hindley (Eds.), Typed Lambda Calculi and Applications, Third International Conference on Typed Lambda Calculi and Applications, TLCA '97, Proceedings, Vol. 1210 of Lecture Notes in Computer Science, Springer, 1997, pp. 196–213.
[41] M. Hasegawa, Models of sharing graphs, Ph.D. thesis, University of Edinburgh (1997).
[42] M. Hyland, Abstract and concrete models for recursion, in: O. Grumberg, T. Nipkow, C. Pfaller (Eds.), Proceedings of the NATO Advanced Study Institute on Formal Logical Methods for System Security and Correctness, IOS Press, 2008, pp. 175–198.
[43] P. Selinger, A survey of graphical languages for monoidal categories, in: B. Coecke (Ed.), New Structures for Physics, Vol. 813 of Lecture Notes in Physics, Springer, 2011, pp. 289–355.
[44] P. M. Hines, The algebra of self-similarity and its applications, Ph.D. thesis, University of Wales, Bangor (1998).
[45] S. Abramsky, E. Haghverdi, P. Scott, Geometry of Interaction and linear combinatory algebras, Mathematical Structures in Computer Science 12 (05) (2002) 625–665.
[46] E. G. Manes, D. B. Benson, The inverse semigroup of a sum-ordered semiring, Semigroup Forum 31 (1) (1985) 129–152.
[47] R. Cockett, R. Garner, Restriction categories as enriched categories, Theoretical Computer Science 523 (2014) 37–55.
[48] M. P. Fiore, Axiomatic domain theory in categories of partial maps, Ph.D. thesis, University of Edinburgh (1994).
[49] P. Selinger, Finite dimensional Hilbert spaces are complete for dagger compact closed categories, Logical Methods in Computer Science 8 (3) (2012) 1–12.

[50] P. Selinger, Idempotents in dagger categories, in: P. Selinger (Ed.), QPL 2006, Vol. 210 of Electronic Notes in Theoretical Computer Science, Elsevier, 2008, pp. 107–122.