Towards Constructive Hybrid Semantics

Tim Lukas Diezel
FAU Erlangen-Nürnberg, Germany
Tim.L.Diezel@fau.de

Sergey Goncharov
FAU Erlangen-Nürnberg, Germany
Sergey.Goncharov@fau.de

Abstract

With *hybrid systems* becoming ever more pervasive, the underlying semantic challenges emerge in their entirety. The need for principled semantic foundations has been recognized previously in the case of discrete computation and discrete data, with subsequent implementations in programming languages and proof assistants. Hybrid systems, contrastingly, do not directly fit into the classical semantic paradigms due to the presence of quite specific "non-programmable" features, such as *Zeno behaviour* and the inherent indispensable reliance on a notion of continuous time. Here, we analyze the phenomenon of hybrid semantics from a constructive viewpoint. In doing so, we propose a monad-based semantics, generic over a given ordered monoid representing the time domain, hence abstracting from the monoid of constructive reals. We implement our construction as a *higher inductive-inductive type* in the recent cubical extension of the Agda proof assistant, significantly using state-of-the-art advances of *homotopy type theory*. We show that classically, i.e. under the *axiom of choice*, our construction admits a charaterization in terms of directed sequence completion.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Axiomatic semantics

Keywords and phrases Hybrid semantics, Elgot iteration, Homotopy type theory, Agda

Digital Object Identifier 10.4230/LIPIcs.FSCD.2020.24

Supplementary Material The recent version of our implementation can be found at `https://github.com/sergey-goncharov/hybrid-agda`.

Funding *Sergey Goncharov*: Support by Deutsche Forschungsgemeinschaft (DFG) under project GO 2161/1-2 is gratefully acknowledged.

Acknowledgements We would like to thank anonymous referees for carefully reading the text and making various points which contributed to improving presentation, and also to Stefan Milius for references on conservative completion.

1 Introduction

Hybrid semantics underlies *cyber-physical systems*, which are systems combining discrete communication and control with continuous evolution of physical (chemical, biological, neuromorphic, etc) processes, typically described by systems of (ordinary) differential equations. Semantic theories, rooted in the classical notion of computability, have been explored thoroughly in recent decades [24, 15, 18]. On the one hand, this has led to a better understanding of the corresponding discrete time systems, thus implicitly or explicitly contributing to improving their design. On the other hand, the results were used in developing verification environments and proof assistants. Hybrid semantics however, requires a massive reconsideration of the established approaches due to a number of features not covered standardly such as *Zeno behaviour*, i.e. the phenomenon of switching the discrete control state infinitely many times within a (physically) finite time interval. Moreover, hybrid computation is inherently intertwined with *reasoning*. For example, in order to describe the movement of a

5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020).
Editor: Zena M. Ariola; Article No. 24; pp. 24:1–24:19
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ball, one has to be able to calculate the moments of collision of the ball with obstacles. From a practical point of view, a clear mathematical formulation of hybrid semantics is needed to be able to deal with verification challenges arising from safety critical systems such as self-driving cars, or aircrafts, or surgeon robots. A natural way to do this is to turn to the *computational*, and more specifically to the *constructive* side of the issue. We thus ask: *"How constructive is hybrid semantics?"* and *"What is the impact of the constructive viewpoint on the verification challenges that may or may not be solved?"*. In order to account for these questions we orient towards principled constructive environments such as *intensional type theory* and the corresponding implementations such as Coq and Agda.

The key concept in the heart of hybrid semantics are *real numbers*. The major existing approaches to hybrid systems rely on classical (non-constructive) real numbers, which are suitable for reasoning but not for computationally feasible operational semantics and not for computer representation of hybrid programs. Observations in a similar vein have been made recently [19, 5]. Here we assume a constructive notion of real numbers and constructive hybrid trajectories, i.e. time indexed sequences, based on them.

In a nutshell, we develop a generalization of the following denotational domain

$$\mathbb{R}_+ \times X \cup \bar{\mathbb{R}}_+ \tag{$\bigstar$}$$

for modelling *durations* of hybrid programs, where $\mathbb{R}_+$ stands for non-negative real numbers and $\bar{\mathbb{R}}_+$ stands for non-negative real numbers extended with infinity. This domain contains pairs (d, x) produced by computations that converge in time d and deliver a final value x, and possibly infinite durations $d\colon \bar{\mathbb{R}}_+$ corresponding to computations that diverge in time d (hence, not delivering any value). We generalize ($\bigstar$) in two directions: by replacing $\mathbb{R}_+$ with a general ordered monoid (to capture various notions of time), and by getting rid of non-constructive principles underlying the use of ($\bigstar$). Roughly, the latter direction is motivated by the fact that ($\bigstar$) does not adequately model iterative computation, unless the *law of excluded middle* is admitted – the fact, that ($\bigstar$) is presented as a disjoint union of denotation domains for convergent and divergent computations, would entail that program termination is *decidable* for this semantics. This phenomenon is known for the *partiality monad* [6, 4], which was developed as a replacement for the *maybe monad* $X \uplus \{\bot\}$, and in exactly the same sense our present construction replaces ($\bigstar$). Even more so, our construction is a direct generalization of the partiality monad construction from [4] and contains the latter as a special case. We dub the obtained monad $\tilde{\mathbf{L}}$ the *(generalized) duration monad* following previous work [9, 10] and keeping in touch with the idea that monoid elements represent time duration, even under a possibly far reaching generalization of the notion of time.

Despite technical similarity, our generalization raises issues which are degenerate and hence ineffective for the partiality monad. For one thing, the idea of characterizations in terms of complete partial orders stems from *domain theory*, where a suitable *information order* for the target denotational domain is assumed, with a bottom element $\bot$ representing divergence. In ($\bigstar$), the notions of divergence continuously range over extended real numbers $\bar{\mathbb{R}}_+$, in particular, we have the least (0) and the greatest (∞) notions of divergence. It turns out that one can select $\bot$ to be 0 and define the information order suitably by combining the standard idea from domain theory, that divergent computations are denotationally smaller than the convergent ones, with the comparison total order on real numbers.

The partiality monad thus becomes the duration monad over a trivial (i.e. single element) ordered monoid, i.e. a computation over it either finishes instantly or diverges. The partial order relation of the trivial ordered monoid is *decidable* in the type-theoretic sense (under the propositions-as-types discipline, the corresponding type satisfies excluded middle). Our paramount example of an ordered monoid though are non-negative reals $\mathbb{R}_+$ whose partial order relation is *not* decidable. The move from the decidable case to possibly undecidable ones has a significant impact on the choice of conditions for countable sequences whose least upper bounds can be computed. Here we stick to *directed sequences* rather than monotone sequences as before [4]. While in the decidable case both approaches are equivalent, in general not only directedness of sequences differs from monotonicity, but also the former comes in two forms: *extensional* and *intensional*. In the latter case, we demand for any two elements of a directed sequence to exist a concrete element greater than both, while in the former one we demand that such an element exists but it is not known which one it is. In type-theoretic terms the difference is expressed by means of *propositional truncation*.

In terms of category theory, we identify each $\tilde{L}X$ as a suitable *free object* on X, or equivalently as an *initial object* in a certain comma category, in accordance with the previous construction of the partiality monad [4]. An alternative approach to constructing the latter presupposes the axiom of countable choice and essentially amounts to quotienting the space of monotone sequence over $X \uplus \{\bot\}$ by weak bisimilarity [6, 21]. This quotienting procedure can also be viewed as *ω-completion* of $X \uplus \{\bot\}$ regarded as a *flat domain* [17]. Under countable choice, both constructions are known to be equivalent. We establish an analogue of this equivalence but only in classical setting (under the full axiom of choice) and, again, replacing monotone sequences with directed ones. Remarkably, the completion procedure gives some insight into Zeno behaviour: when forming the completion of a partial order, Zeno behaviour contributes via duplication of least upper bounds that already exist in the original set because those cannot be detected via the completion procedure. For example, directed sequence completion (in contrast e.g. to *Cauchy completion*) cannot identify the sequence $1/2, 3/4, 7/8, \ldots$ and $1, 1, 1, \ldots$ To remedy this, we also introduce a coarsened version $\bar{\mathbf{L}}$ of $\tilde{\mathbf{L}}$ by additionally demanding that all the originally existing least upper bounds must be kept intact. This corresponds to a modified completion procedure [14], which is dubbed *conservative completion* in [23].

We formalized our duration monad via higher inductive-inductive types in the Agda proof assistant (version 2.6.1-96d0dd0) using the version of the cubical library from Feb 6 2020. The recent version of our implementation can be found at `https://github.com/sergey-goncharov/hybrid-agda`.

Paper Organization

After short preliminaries in Section 2, we present our motivation in Section 3. In Section 4 we provide our main construction using complete monoid modules in categorical terms, and subsequenty characterize the obtained object in Section 5 under the assumption of the axiom of choice. In Section 6 we give the main construction of the generalized duration monad coping with Zeno behaviour. We discuss our formalization of both our constructions as higher inductive-inductive types in cubical Agda in Section 7. A conclusion and our plans of further work are given in Section 8.

2 Preliminaries

We work in an ambient theory of sets **Set** throughout, unless stated otherwise not assuming it to validate either excluded middle or any form of choice. For example **Set** can be understood as the type of HoTT types of h-level 2 [20]. Generally, we refer to the cited *HoTT book* as a comprehensive presentation of the underlying foundational realm for our results. An *ordered monoid* is a monoid $(\mathbb{M}, +, 0)$ together with a partial order $\leqslant$ on $\mathbb{M}$ such that 0 is the least element and $+$ is monotone on the right, i.e. $0 \leqslant a$ and $b \leqslant c \Rightarrow a + b \leqslant a + c$ for all $a, b, c \colon \mathbb{M}$.

We do not generally assume the dual monotonicity law (!) $a \leqslant b \Rightarrow a + c \leqslant b + c$. Even though we use the additive notation $+$ for generic monoids, we do not assume commutativity of $+$. By $\mathbb{R}$, $\mathbb{R}_+$ and $\bar{\mathbb{R}}_+$ we respectively denote (constructive) reals, non-negative reals and extended non-negative reals (i.e. $\mathbb{R}_+$ suitably extended with the infinite value ∞). For least upper bounds of families $(s_i)_i$, we interchangeably use the notation $\bigvee(s_i)_i$ and $\bigvee_i s_i$.

We assume basic familiarity with the concepts of category theory, specifically with *universal arrows* [13] in the form of *free objects* [2] (i.e. universal arrows for faithful functors). A monad $\mathbf{T}$ (on $\mathbf{Set}$) is determined by a *Kleisli triple* $(T, \eta, (-)^\star)$, consisting of a map $T\colon \mathbf{Set} \to \mathbf{Set}$, together with a $\mathbf{Set}$-indexed class of morphisms $\eta_X\colon X \to TX$ and *Kleisli lifting* sending each $f\colon X \to TY$ to $f^\star\colon TX \to TY$ and obeying *monad laws*: $\eta^\star = \mathsf{id}$, $f^\star\eta = f$, $(f^\star g)^\star = f^\star g^\star$ (it follows from this definition that T extends to a functor and η to a natural transformation). The map $f, g \mapsto f^\star g$ is called *Kleisli composition.* In program semantics, in order to interpret while-loops, one more specifically needs monads equipped with a notion of iteration. Monads from a suitable class called *(complete) Elgot monads* are required to support an *Elgot iteration* operator $(f\colon X \to T(Y \uplus X)) \mapsto (f^\dagger\colon X \to TY)$ subject to established laws of iteration [11]. We will continue to use bold capitals (e.g. $\mathbf{T}$) for monads over the corresponding endofunctors written as capital Romans (e.g. T).

3 Hybrid Semantics and Beyond

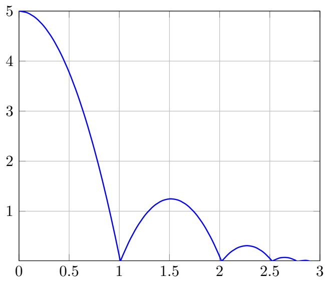
Let us briefly recall the hybrid language HybCore from previous work [10]. The grammar is as follows:

$$\begin{aligned} v, w &::= x \mid \star \mid \mathbf{true} \mid \mathbf{false} \mid (v, w) \mid f(v) && (f \in \Sigma) \\ p, q &::= \lceil v \rceil \mid (x, y) := p;\, q \mid x := p;\, q \\ &\quad\mid x := t.\, v \;\&\; w \mid \mathbf{if}\ v\ \mathbf{then}\ p\ \mathbf{else}\ q \mid x := p\ \mathbf{while}\ v\ \{q\} \end{aligned}$$

Here, x, y refer to variables, v, w refer to *values* and p, q refer to *programs* (as prescribed by the *fine-grain call-by-value* discipline [12]). Besides the standard Boolean values (**true** and **false**) and the canonical value $\star$ of the unit type, new values can be generated by Cartesian pairs (v, w), and by transforming other values with functions coming from a custom signature Σ. The latter is specifically meant to contain all the necessary parametrized time-dependent functions of type $X \times \mathbb{R}_+ \to Y$ representing continuous dynamics, and are regarded as atomic constructs by HybCore. A program either instantly returns a value ($\lceil v \rceil$), or is obtained by using one of the standard imperative style constructs, or by using the construct $x := t.\, v \;\&\; w$, which simultaneously abstracts v over the time variable t and restricts the domain of definiteness of the obtained function to the largest interval (either of the form $[0, d)$ or $[0, d]$) on which the predicate w holds throughout. A standard example is *bouncing ball*:

$$x := \lceil(5, 0)\rceil\ \mathbf{while\ true}\ \{(h, v) := (x := t.\, \mathit{ball}(x, t) \;\&\; \mathsf{fst}\, x \geqslant 0);\, \lceil(h, -0.5v)\rceil\}$$

Here, the height h and the velocity v are initially set to 5 and 0 correspondingly, and *ball* is a signature symbol representing the continuous dynamics of a flying ball. Each time the ball touches the ground v is reset to $-0.5v$ with -0.5 standing for the *damping factor.* This behaviour is repeated in the loop indefinitely. Graphically, we obtain the following *trajectory*



representing the dependency of the ball's hight in time. Among other things, this example demonstrates *Zeno behaviour* which is inherent to hybrid systems: infinitely many iterations occur in finite time.

We use HybCore as a motivation for the *duration semantics*, which assigns to any program a map $f\colon X \to \mathbb{R}_+ \times Y \cup \bar{\mathbb{R}}_+$ meaning that

- if $f(x) = (d, y)\colon \mathbb{R}_+ \times Y$ then the program yields the final value y in finite time d;
- if $f(x) = d \in \bar{\mathbb{R}}_+$ then the program diverges in finite or infinite time d (e.g. by exposing Zeno behaviour) and hence does not deliver any final value.

This semantics yields a monad $\mathbb{R}_+ \times (-) \cup \bar{\mathbb{R}}_+$, which is further generalized as follows.

▶ **Definition 1** (Monoid Module, Generalized Writer Monad [10]). *Given a monoid* $(\mathbb{M}, +, 0)$, *a* monoid module *is a set* $\mathbb{E}$ *equipped with a map* $\rhd : \mathbb{M} \times \mathbb{E} \to \mathbb{E}$, *subject to the laws*

$$0 \rhd e = e \qquad\qquad (m + n) \rhd e = m \rhd (n \rhd e)$$

Every monoid-module pair $(\mathbb{M}, \mathbb{E})$ *induces the following monad* $\mathbf{T} = (T, \eta, (-)^\star)$ *which we call the* generalized writer monad*:* $T = \mathbb{M} \times (-) \cup \mathbb{E}$, $\eta_X(x) = (0, x)$, *and*

$$\begin{aligned} f^\star(m, x) &= (m + n, y) && \textit{where} \quad m\colon \mathbb{M},\ x\colon X,\ f(x) = (n, y)\colon \mathbb{M} \times Y \\ f^\star(m, x) &= m \rhd e && \textit{where} \quad m\colon \mathbb{M},\ x\colon X,\ f(x) = e\colon \mathbb{E} \\ f^\star(e) &= e && \textit{where} \quad e\colon \mathbb{E} \end{aligned}$$

This yields a joint generalization of the writer monad $(\mathbb{E} = \varnothing)$ *and the exception monad* $(\mathbb{M} = 1)$.

In order to interpret while-loops of HybCore w.r.t. the duration semantics, one needs to turn $\mathbb{R}_+ \times (-) \cup \bar{\mathbb{R}}_+$ into an Elgot monad, which is, however, impossible in a constructive setting because this would imply decidability of program divergence. This is analogous to the fact that the maybe-monad, i.e. the generalized writer monad over $\mathbb{M} = 1$, $\mathbb{E} = 1$ cannot generally serve as a model of partiality [4].

We next abstract from a concrete choice of $\mathbb{M}$ and fix the following stock of running examples for further reference.

▶ **Example 2** (Ordered Monoids). Following are some ordered monoids on **Set**.

1. $(1, !, \star, \{(\star, \star)\})$ is a trivial ordered monoid over a one-element carrier $1 = \{\star\}$.
2. $(\mathbb{N}, +, 0, \leqslant)$ is an ordered monoid of natural numbers $\mathbb{N}$.
3. $(\mathbb{Q}_+, +, 0, \leqslant)$ is an ordered monoid of non-negative rational numbers $\mathbb{Q}_+$.
4. $(\mathbb{R}_+, +, 0, \leqslant)$ is an ordered monoid of non-negative real numbers $\mathbb{R}_+$.
5. $(A^\star, \cdot, \epsilon, \leqslant)$ is an ordered monoid of finite strings where ϵ is the empty string and $\leqslant$ is defined as follows: $u \leqslant v$ iff u is a prefix of v, i.e. there exists a w such that $uw = v$.

6. $(A^{[0,\mathbb{R}_+)}, {}^\frown, (0,!), \leqslant)$, the *monoid of (finite) trajectories* over a given set A, is defined as follows: $A^{[0,\mathbb{R}_+)} = \Sigma_{d\colon \mathbb{R}_+}\, [0,d) \to A$ is the set of *finite trajectories* valued on A; elements are pairs (d,e) where $d\colon \mathbb{R}_+$ and $e\colon [0,d) \to A$ is a *trajectory* of duration d; the concatenation operation ${}^\frown$ is defined as follows:
$$(d_1,e_1) {}^\frown (d_2,e_2) = (d_1 + d_2, \lambda t.\, \textit{if } t < d_1 \textit{ then } e_1(t) \textit{ else } e_2(t-d_1)).$$
The unit is $(0,!)$ where $!\colon [0,0) \to A$ is the *empty trajectory* and the relation $\leqslant$ is defined as follows: $(d_1,e_1) \leqslant (d_2,e_2)$ if $d_1 \leqslant d_2$ and $e_1(t) = e_2(t)$ for every $t\colon [0,d_1)$. Note that $A^\star$ is isomorphic to $\Sigma_{n\colon \mathbb{N}}\, A^n$, hence $A^{[0,\mathbb{R}_+)}$ can be understood as a counterpart of $A^\star$ obtained by changing the underlying notion of time from discrete ($\mathbb{N}$) to continuous ($\mathbb{R}_+$).
7. $(L, \vee, 0, \leqslant)$ is an ordered monoid for any join semilattice L with a bottom element 0.
8. $(\mathbb{M}_1 \times \mathbb{M}_2, +, (0_1,0_2), \leqslant)$ is an ordered monoid, provided that so are $(\mathbb{M}_1, +_1, 0_1, \leqslant_1)$ and $(\mathbb{M}_2, +_2, 0_2, \leqslant_2)$; under these assumptions $(a_1,a_2) + (b_1,b_2) = (a_1 +_1 b_1, a_2 +_2 b_2)$ and $(a_1,a_2) \leqslant (b_1,b_2)$ if $a_1 \leqslant_1 b_1$ and $a_2 \leqslant_2 b_2$.

We intuitively regard ordered monoids as carriers of various (possibly exotic) notions of time. The examples **4.** and **6.** are of direct use for hybrid semantics: the former (*duration semantics*) corresponds to the semantics capturing only durations of programs; the latter (*evolution semantics*) captures both the durations and the intermediate states arranged in trajectories (cf. [10]). The monoids in **2.** and **3.** capture discrete and rational notions of time. In **5.**, the discrete time instants are labelled by the elements of A, in particular, $1^\star \cong \mathbb{N}$ corresponds to vacuous labelling. Both **5.** and **6.** illustrate our decision to make do without the left monotonicity law $a \leqslant b \Rightarrow a + c \leqslant b + c$, which is not satisfied by these examples.

We will treat an ordered monoid $\mathbb{M}$ as an input parameter to our constructions, while the corresponding monoid module $\mathbb{E}$ will universally arise from $\mathbb{M}$. In contrast to the generalized writer monad, $\mathbb{E}$ will no longer be a disjoint component of TX, however, the equation $T\emptyset = \mathbb{E}$ will have to remain true. In fact, the perspective we take is to consider the whole TX as a monoid module, regarded as an algebraic structure, and generated by X.

4 Complete Monoid Modules, Categorically

To obtain a constructively feasible replacement for the monad in Definition 1, we follow the idea used to define the partiality monad in intensional type theory [4], which can be abstractly summarized as follows:

1. Introduce a suitable category of algebras $\mathbf{Alg}_{\mathbf{T}}$ over $\mathbf{Set}$.
2. Obtain $\mathbf{T}$ from an adjunction $\begin{tikzcd} \mathbf{Set} \arrow[r, bend left] \arrow[r, phantom, "\perp"] & \mathbf{Alg}_{\mathbf{T}} \arrow[l, bend left] \end{tikzcd}$.

This scenario raises three main questions:

- How to specify the category of algebras $\mathbf{Alg}_{\mathbf{T}}$?
- How to construct $\mathbf{T}$ (i.e. prove that it exists)?
- How to ensure that the constructed monad $\mathbf{T}$ does indeed faithfully capture the intended semantics?

In this section we introduce a monad $\tilde{\mathbf{L}}$ parametrized by a generic ordered monoid $\mathbb{M}$. The construction of $\tilde{\mathbf{L}}$ is formulated in abstract category-theoretic terms, thus remaining agnostic about any specific choice of foundations. In Section 5, we then show that classically $\tilde{\mathbf{L}}$ can be characterized in terms of directed sequence completion and in Section 7 we discuss a formalization of $\tilde{\mathbf{L}}$ in the constructive realm of HoTT and cubical Agda.

As the first step, we identify the category of algebras $\mathbf{Alg}_{\tilde{\mathbf{L}}}$, which in our case are called *complete* $\mathbb{M}$*-modules*. Complete $\mathbb{M}$-modules are a proper generalization of *partiality algebras* [4] (corresponding to trivial $\mathbb{M}$).

▶ **Definition 3** (Complete $\mathbb{M}$-Modules). *An* ordered $\mathbb{M}$-module *w.r.t. an ordered monoid* $(\mathbb{M}, +, 0, \leqslant)$*, is an* $\mathbb{M}$*-module* $(\mathbb{E}, \triangleright)$ *together with a partial order* $\sqsubseteq$ *and a least element* $\bot$*, such that* $\triangleright$ *is monotone on the right and* $(- \triangleright \bot)$ *is monotone, i.e.*

$$\frac{}{\bot \sqsubseteq x} \qquad\qquad \frac{x \sqsubseteq y}{a \triangleright x \sqsubseteq a \triangleright y} \qquad\qquad \frac{a \leqslant b}{a \triangleright \bot \sqsubseteq b \triangleright \bot}$$

We call the last property restricted left monotonicity*. An infinite sequence* $s_1, s_2, \ldots$ *is* monotone *if* $s_i \sqsubseteq s_{i+1}$ *for every* i *and* directed *if for every* i *and every* j *there exists* k *such that* $s_i \sqsubseteq s_k$ *and* $s_j \sqsubseteq s_k$*. Clearly, every monotone sequence is directed.*

An ordered $\mathbb{M}$*-module is* (directed ω-)complete *if for every directed sequence* $(s_i)_i$ *on* $\mathbb{E}$ *there is a least upper bound* $\bigsqcup_i s_i$ *and* $\triangleright$ *is continuous on the right, i.e.*

$$\frac{}{s_i \sqsubseteq \bigsqcup_i s_i} \qquad\qquad \frac{\forall i.\, s_i \sqsubseteq x}{\bigsqcup_i s_i \sqsubseteq x} \qquad\qquad \frac{}{\bigsqcup_i a \triangleright s_i \sqsubseteq a \triangleright \bigsqcup_i s_i}$$

(the law $a \triangleright \bigsqcup_i s_i \sqsubseteq \bigsqcup_i a \triangleright s_i$ *is derivable).*

Note that any ordered monoid $\mathbb{M}$, which is complete as a partial order is a complete $\mathbb{M}$-module under $\triangleright = +$, provided that $+$ is right continuous. Consider further examples of complete $\mathbb{M}$-modules.

▶ **Example 4.** Let us revisit Example **2.** For illustration purposes, let us assume here that **Set** is a classical set theory, e.g. ZFC.

- In **2.**–**4.**, $\bar{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$ is an ordered $\mathbb{N}$-module, $\bar{\mathbb{Q}}_+ = \mathbb{Q}_+ \cup \{\infty\}$ is an ordered $\mathbb{Q}_+$-module and $\bar{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{\infty\}$ is an ordered $\mathbb{R}_+$-module respectively: $x \triangleright y = x + y$ and $x \triangleright \infty = \infty$, $\bot = 0$, $x \leqslant y \Rightarrow x \sqsubseteq y$ and $x \sqsubseteq \infty$ always. Both $\bar{\mathbb{Q}}_+$ and $\bar{\mathbb{R}}_+$ are also ordered $\mathbb{N}$-modules and $\bar{\mathbb{R}}_+$ is an ordered $\mathbb{Q}_+$-module. Now, $\bar{\mathbb{N}}$ is a complete $\mathbb{N}$-module and $\bar{\mathbb{R}}_+$ is a complete $\mathbb{R}_+$-module: $\bigsqcup_i s_i$ is the least upper bound of $(s_i)_i$ if s is bounded and ∞ otherwise. Also $\bar{\mathbb{R}}_+$ is a complete $\mathbb{N}$-module and a complete $\mathbb{Q}_+$-module, but $\bar{\mathbb{Q}}_+$ is not complete w.r.t. any monoid because it is not complete as a partial order.
- In **5.**, the set A^ω of infinite strings and the set $A^{\leqslant\omega} = A^\star \cup A^\omega$ of finite and infinite strings are both $A^\star$-modules under prefixing a string with a finite string. Moreover, $A^{\leqslant\omega}$ is a complete $A^\star$-module with the empty word as $\bot$ and least upper bounds calculated in the obvious way.
- In **6.** we defined the set $A^{[0,\mathbb{R}_+)} = \Sigma_{d\colon \mathbb{R}_+} [0, d) \to A$ of finite trajectories, which is of course an $A^{[0,\mathbb{R}_+)}$-module on itself. Analogously, let $A^{[0,\bar{\mathbb{R}}_+)} = \Sigma_{d\colon \bar{\mathbb{R}}_+} [0, d) \to A$ be the set of *finite or infinite trajectories* under the same operations, partially ordered in the same way and with $\triangleright$ extended as follows: $(d_1, e_1) \triangleright (d_2, e_2) = (d_1, e_1) \frown (d_2, e_2)$ if $d_1\colon \mathbb{R}_+$ and $(\infty, e_1) \triangleright (d_2, e_2) = (\infty, e_1)$. Now, to calculate the least upper bound of a directed sequence $(d_1, e_1), (d_2, e_2), \ldots$, we first calculate $d = \bigvee_i d_i$ and then calculate $e\colon [0, d) \to A$ at each point $x\colon [0, d)$ by setting $e(x) = e_i(x)$ for sufficiently large i, which is guaranteed to exist by definition. Thus $A^{[0,\bar{\mathbb{R}}_+)}$ is a complete $A^{[0,\mathbb{R}_+)}$-module.

We emphasize the non-constructive flavor of the above examples: when forming a disjoint union such as $A^\star \cup A^\omega$, we make an explicit distinction between finite and infinite data, which cannot be realized constructively. For example, it is not possible to decide whether a string is finite or infinite on the basis of a given finite prefix whatever long.

Complete $\mathbb{M}$-modules form a category $\mathbf{Alg}_{\tilde{\mathsf{L}}}$ together with complete $\mathbb{M}$-module morphisms, which we define as follows.

▶ **Definition 5** (Complete $\mathbb{M}$-Module Morphisms). *Given two complete $\mathbb{M}$-modules $\mathbb{E}$ and $\mathbb{F}$, a* complete $\mathbb{M}$-module morphism *from $\mathbb{E}$ to $\mathbb{F}$ is a map $f\colon \mathbb{E} \to \mathbb{F}$ such that*

$$\frac{\qquad}{f(a \rhd x) = a \rhd f(x)} \qquad \frac{\qquad}{f(\bot) = \bot} \qquad \frac{x \sqsubseteq y}{f(x) \sqsubseteq f(y)} \qquad \frac{\qquad}{f\big(\bigsqcup_i s_i\big) = \bigsqcup_i f(s_i)}$$

We next instantiate the general categorical notion of *free object* [2] to complete $\mathbb{M}$-modules as follows.

▶ **Definition 6** (Free Complete $\mathbb{M}$-Modules). *A* free complete $\mathbb{M}$-module *on a set X consists of a complete $\mathbb{M}$-module $\widetilde{L}X$ and a map $\eta_X\colon X \to \widetilde{L}X$, such that for every map $f\colon X \to \mathbb{E}$ there exists a unique complete $\mathbb{M}$-module morphism $f^\star\colon \widetilde{L}X \to \mathbb{E}$ and the following diagram commutes*

$$\begin{tikzcd}
\widetilde{L}X \arrow[r, dotted, "f^\star"] & \mathbb{E} \\
X \arrow[u, "\eta_X"] \arrow[ur, "f"'] &
\end{tikzcd} \tag{1}$$

In different terms, $(\widetilde{L}X, \eta_X)$ is an initial object in the comma category $X \downarrow \eta_X$.

We proceed under the assumption that every $\widetilde{L}X$ exists and defer further details related to this issue until Section 7 where we discuss our construction of $\widetilde{L}X$ as a quotient inductive-inductive type. For now, we concentrate on checking if the object $\widetilde{L}X$ does indeed correctly capture the intended semantics. First, observe the following.

▶ **Theorem 7.**

1. *The forgetful functor $U\colon \mathbf{Alg}_{\widetilde{\mathsf{L}}} \to \mathbf{Set}$ has a left adjoint $F\colon \mathbf{Set} \to \mathbf{Alg}_{\widetilde{\mathsf{L}}}$ inducing a monad over $\widetilde{L} = UF$ with unit η and Kleisli lifting $(\text{–})^\star$ agreeing with* (1)*.*
2. *$\widetilde{\mathsf{L}}$ is enriched over directed complete partial orders, and moreover, Kleisli composition is strict on both sides.*
3. *$\widetilde{\mathsf{L}}$ is an Elgot monad with the iteration operator $(f\colon X \to \widetilde{\mathsf{L}}(Y \uplus X))^\dagger$ calculated as a least fixed point of the map $[\eta, \text{–}]^\star f\colon (X \to \widetilde{L}Y) \to (X \to \widetilde{L}Y)$.*

Proof.

1. It is a standard category-theoretic fact [2] that existence of all free objects $\widetilde{L}X$ implies existence of the left adjoint F to the forgetful functor U. The arising monad $\widetilde{\mathsf{L}}$ is then as described.

2. Every $\widetilde{L}X$ carries a complete partial order by definition. The laws $f^\star(\bot) = \bot$, $p \sqsubseteq q \Rightarrow f^\star(p) \sqsubseteq f^\star(q)$ and $f^\star\big(\bigsqcup_i s_i\big) = \bigsqcup_i f^\star(s_i)$ follow from the fact that $f^\star$ is a complete $\mathbb{M}$-module morphism by definition. The dual properties of Kleisli composition amount to $f \sqsubseteq g \Rightarrow f^\star \sqsubseteq g^\star$ and $\bigsqcup_i f_i^\star = \big(\bigsqcup_i f_i\big)^\star$ assuming pointwise extension of the order on the function spaces. Assuming the former, the latter easily follows from the universal property (1) since $\big(\bigsqcup_i f_i^\star\big)\,\eta = \bigsqcup_i f_i^\star\,\eta = \bigsqcup_i f_i = \big(\bigsqcup_i f_i\big)^\star\eta$.

The fact that $f \sqsubseteq g$ implies $f^\star \sqsubseteq g^\star$ (for $f, g\colon X \to \widetilde{L}Y$) is by no means entailed by a generic category-theoretic argument. We show it as follows in slightly more generality for any $f, g\colon X \to \mathbb{E}$. For any complete $\mathbb{M}$-module $\mathbb{E}$, let $\mathbb{E}^{\downarrow}$ be the set of down-closed $\mathbb{E}$-submodules of $\mathbb{E}$. Then $\mathbb{E}^{\downarrow}$ is itself an $\mathbb{E}$-module under

$$a \rhd S = \{a \rhd x \mid x\colon S\}, \qquad \bot = \varnothing, \qquad S \sqsubseteq R \iff S \subseteq R, \qquad \bigsqcup_i S_i = \bigcup_i S_i.$$

Now, given $f, g\colon X \to \mathbb{E}$, let $f{\downarrow}$ and $g{\downarrow}$ be the pointwise principal ideals induced by f and g, e.g. $f{\downarrow}(x) = \{y \mid y \sqsubseteq f(x)\}$. Now, if $f \sqsubseteq g$ then $(f^\star{\downarrow} \cup g^\star{\downarrow})\,\eta = f{\downarrow} \cup g{\downarrow} = g{\downarrow} = g^\star{\downarrow}\,\eta$, hence $f^\star{\downarrow} \cup g^\star{\downarrow} = g^\star{\downarrow}$, which entails the desired inequation $f^\star \sqsubseteq g^\star$.

3. Finally, the fact that the above defined iteration operator turns $\tilde{\mathbf{L}}$ into an Elgot monad was shown in previous work [11]. ◀

▶ Remark 8. Observe that $\mathbf{Alg}_{\tilde{\mathbf{L}}}$ need not be the Eilenberg-Moore category of $\tilde{\mathbf{L}}$. Already with $\mathbb{M} = 1$ we obtain as $\mathbf{Alg}_{\tilde{\mathbf{L}}}$ the category of directed complete partial orders, which is known not to be monadic over **Set**.

Using Theorem 7, we can immediately equip HybCore with a denotational semantics. The key clause, not entailed by the standard monad-based paradigm [16, 12], is while-loops, for which we put

$$\begin{aligned}&[\![x := p \ \textbf{while}\ b\ \{q\}]\!](\sigma)\\ &\qquad = \big((\lambda\sigma, x.\, \mathit{if}\ [\![b]\!](\sigma, x)\ \mathit{then}\ (\tilde{L}\,\mathsf{inr})[\![q]\!](\sigma, x)\ \mathit{else}\ \eta(\mathsf{inl}\,x))^\dagger\big)^\star(\sigma, [\![p]\!]\sigma)\end{aligned}$$

where $\sigma\colon [\![\Gamma]\!]$ interprets free variables from the variable context Γ, $[\![p]\!]\colon [\![\Gamma]\!] \to \tilde{L}[\![X]\!]$ and $[\![q]\!]\colon [\![\Gamma]\!] \times [\![X]\!] \to \tilde{L}[\![X]\!]$ are semantics of the involved programs, and $[\![b]\!]\colon [\![\Gamma]\!] \times [\![X]\!] \to 2$ is the semantics of the Boolean test b. By applying this to Example 2 (4) and to Example 2 (6), we obtain duration semantics and evolution semantics correspondingly (cf. [10]).

5 Complete Monoid Modules, Classically

Throughout this section we assume that **Set** is a classical set theory satisfying the axiom of choice (and hence also the law of excluded middle). Some portion of the presented developments can in fact be interpreted constructively or using more conservative principles such as the axiom of countable choice. However, we currently do not know how to fully rebase the following material on such more relaxed assumptions.

Our purpose here is to obtain a concrete description of the monad $\tilde{\mathbf{L}}$ and make sure that the result is in agreement with the expectations. To that end we develop an alternative characterization of $\tilde{L}X$ in terms of directed sequences.

▶ **Definition 9** (Directed Sequences). *A directed sequence is a countable infinite sequence $s_1, s_2, \ldots$ with the property that for every i and every j there is k such that $s_i \sqsubseteq s_k$ and $s_j \sqsubseteq s_k$. An (ω-)directed complete set is a partially ordered set in which every (ω-)directed sequence has a least upper bound.*

We start off by equipping the set $\mathbb{M}_X = \mathbb{M} \times (X \uplus \{\bot\})$ with the structure of an ordered $\mathbb{M}$-module as follows:

$$\frac{}{a \rhd_X (b, x) = (a + b, x)} \qquad \frac{}{(a, \mathsf{inl}\,x) \sqsubseteq_X (a, \mathsf{inl}\,x)} \qquad \frac{a \leqslant b}{(a, \mathsf{inr}\,\bot) \sqsubseteq_X (b, x)}$$

The idea is to use the elements of directed sequences $s\colon \mathbb{N} \to \mathbb{M}_X$ as progressively improving pieces of information about the final outcome of the underlying computational process. The elements of the form $(a, \mathsf{inl}\,x)$ are the maximal elements of this order, indicating that x is a final output of the process at time instant a. The elements of the form $(a, \mathsf{inr}\,\bot)$ represent *potential divergence* the time instant a, or later. The constraint $(a, \mathsf{inr}\,\bot) \sqsubseteq_X (b, x)$ with $a \leqslant b$ indicates that this potential divergence can still be resolved into successful termination over time. However, this need not happen. In particular, we allow for Zeno behaviour in the form of monotone sequences:

$$(a_1, \mathsf{inr}\,\bot) \sqsubseteq_X (a_2, \mathsf{inr}\,\bot) \sqsubseteq_X \dots$$

with $a_1 < a_2 < \dots$ where the set $\{a_1, a_2, \dots\}$ has an upper bound in $\mathbb{M}$. Existence of such sequences, of course, depends on the properties of $\mathbb{M}$. For example, they do not exist for $\mathbb{M} = \mathbb{N}$, but they do exist for $\mathbb{R}_+$, e.g. $a_1 = 1/2$, $a_2 = 1/2 + 1/4$, ...

The following property is easy to verify.

▶ **Proposition 10.** *For any set X, $(\mathbb{M}_X, \triangleright_X, (0, \mathsf{inr}\,\bot), \sqsubseteq_X)$ is an ordered $\mathbb{M}$-module.*

We next need to quotient the space of directed sequences $\mathbb{N} \to \mathbb{M}_X$ suitably to ensure that two sequences which tend to the same value are indistinguishable. To this end, we adapt the standard idea of chain completion from domain theory [17] by moving from monotone sequences to directed sequences.

▶ **Definition 11** (Directed Sequence Completion). *For any poset $(A, \leqslant)$, we define a preorder $\lesssim$ on directed sequences over A as follows:*

$$(s_i)_i \lesssim (t_i)_i \iff \forall i \colon \mathbb{N}.\, \exists j \colon \mathbb{N}.\, s_i \leqslant t_j.$$

This induces the equivalence $\sim$ on directed sequences as follows:

$$s \sim t \iff s \lesssim t \wedge t \lesssim s.$$

The directed sequence completion *$\widetilde{A}$ of A is the poset $(\widetilde{A}, \lessapprox)$, defined as follows: $\widetilde{A}$ is the quotient of the space of directed sequences over A by $\sim$, and $[s]_\sim \lessapprox [t]_\sim$ whenever $s \lesssim t$, where, as usual, we denote by $[s]_\sim \colon \widetilde{A}$ the equivalence class of $s \colon \mathbb{N} \to A$ in $\widetilde{A}$. Let us also agree to use the notation $[s_i]_i$ instead of $[(s_i)_i]_\sim$.*

▶ Remark 12 (Ideal Completion). An apparently more common, and classically equivalent, way (see e.g. [23, 1]) to introduce $\widetilde{A}$ is to use *ideal completion*. An ideal in A is a nonempty subset of A that is downward closed and directed. We could thus alternatively view $\widetilde{A}$ not as a set of equivalence classes but as a set of ideals of A generated by directed sequences. This switch of perspective is based on a representation of quotients by means of equivalence classes, which is uncomplicated for set theory but, of course, not for type theory.

The directed sequence completion of $\mathbb{M}_X$ yields a complete $\mathbb{M}$-module $\widetilde{\mathbb{M}}_X$. To show this we use the *Cantor pairing function* $\pi \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with

$$\pi(x, y) = \frac{1}{2}(x + y)(x + y + 1) + x,$$

which witnesses an isomorphism between $\mathbb{N}$ and $\mathbb{N} \times \mathbb{N}$ whose inverse we denote $\pi^{-1} = (\pi_1^{-1}, \pi_2^{-1}) \colon \mathbb{N} \to \mathbb{N} \times \mathbb{N}$.

▶ **Proposition 13.** *The quotient $(\widetilde{\mathbb{M}}_X, \triangleright, \bot, \lessapprox, \overset{\approx}{\bigvee})$ is a complete $\mathbb{M}$-module under*

$$a \triangleright [s_i]_i = [a \triangleright_X s_i]_i, \qquad \bot = [(0, \mathsf{inr}\,\bot)]_i, \qquad \overset{\approx}{\bigvee}_i [s_{i,j}]_j = [s_{\pi_1^{-1}(i), \pi_2^{-1}(i)}]_i.$$

Proof Sketch. If $(s_i)_i \sim (t_i)_i$ for directed sequences $(s_i)_i$ and $(t_i)_i$ then it is easy to see that both $(a \triangleright_X s_i)_i$ and $(a \triangleright_X t_i)_i$ are directed and $(a \triangleright_X s_i)_i \sim (a \triangleright_X t_i)_i$, hence $\triangleright$ is correctly defined. Let us check that $\overset{\approx}{\bigvee}_i [s_{i,j}]_j$ is correctly defined. First, if $(s_{i,j})_j \sim (t_{i,j})_j$ for every i, then for all j there is j' such that $s_{i,j} \leqslant t_{i,j'}$ for every i, and conversely for all j there is j' such that $t_{i,j} \leqslant s_{i,j'}$ for every i, hence $(s_{\pi_1^{-1}(i),\pi_2^{-1}(i)})_i \sim (t_{\pi_1^{-1}(i),\pi_2^{-1}(i)})_i$. Therefore, the expression for $\overset{\approx}{\bigvee}_i\,[s_{i,j}]_j$ does not depend on the representatives of the equivalence classes $[s_{i,j}]_i$. To show correctness of the definition of $\overset{\approx}{\bigvee}_i\,[s_{i,j}]_j$, we are left to verify that $(s_{\pi_1^{-1}(i),\pi_2^{-1}(i)})_i$ is directed. Let $n, m\colon \mathbb{N}$ and construct a suitable k such that $s_{\pi_1^{-1}(n),\pi_2^{-1}(n)} \leqslant s_{\pi_1^{-1}(k),\pi_2^{-1}(k)}$ and $s_{\pi_1^{-1}(m),\pi_2^{-1}(m)} \leqslant s_{\pi_1^{-1}(k),\pi_2^{-1}(k)}$. Let $(n_1, n_2) = \pi^{-1}(n)$ and $(m_1, m_2) = \pi^{-1}(m)$. By assumptions, there is k_1 such that $(s_{n_1,j})_j \lesssim (s_{k_1,j})_j$ and $(s_{m_1,j})_j \lesssim (s_{k_1,j})_j$. Thus we obtain $n_2', m_2'\colon \mathbb{N}$ such that $s_{n_1,n_2} \leqslant s_{k_1,n_2'}$ and $s_{m_1,m_2} \leqslant s_{k_1,m_2'}$ and $k_2\colon \mathbb{N}$ such that $s_{n_1,n_2} \leqslant s_{k_1,n_2'} \leqslant s_{k_1,k_2}$ and $s_{m_1,m_2} \leqslant s_{k_1,m_2'} \leqslant s_{k_1,k_2}$. Hence, we can take $k = \pi(k_1, k_2)$.

The axioms of complete $\mathbb{M}$-modules then transfer from $\mathbb{M}_X$ to $\widetilde{\mathbb{M}}_X$. ◀

The defined $\mathbb{M}$-module is in fact the free one on X.

▶ **Theorem 14.** *$((\widetilde{\mathbb{M}}_X, \rhd, \bot, \widetilde{\lesssim}, \overset{\approx}{\bigvee}), \eta)$ is the free complete $\mathbb{M}$-module on X with $\eta(x) = [(0, \mathsf{inl}\, x)]_i$.*

Proof Sketch. We fix a complete $\mathbb{M}$-module $(\mathbb{E}, \rhd, \bot, \sqsubseteq, \bigsqcup)$ together with a map $f\colon X \to \mathbb{E}$. Our goal is to construct a unique complete $\mathbb{M}$-module morphism $f^\star\colon \widetilde{\mathbb{M}}_X \to \mathbb{E}$ satisfying $f^\star(\eta(x)) = f(x)$. To this purpose we define an auxiliary function $\ell\colon X \uplus \{\bot\} \to \mathbb{E}$ as follows:

$$\ell(\mathsf{inl}\, x) = f(x), \qquad\qquad \ell(\mathsf{inr}\,\bot) = \bot.$$

We then define $f^\star\colon \widetilde{\mathbb{M}}_X \to \mathbb{E}$ by putting $f^\star[(a_i, x_i)]_i = \bigsqcup_i a_i \rhd \ell(x_i)$. The remaining technical work amounts to showing that the above definition of $f^\star$ is valid and that $f^\star$ is the unique morphism making (1) commute. ◀

Theorem 14 can be understood as a soundness and completeness property: it shows that the axioms of complete $\mathbb{M}$-modules are sound and complete over the models arising from $\mathbb{M}$ by directed sequence completion. Hence, we also obtain a more explicit description of the object $\widetilde{L}X$.

▶ **Corollary 15.** *$\widetilde{L}X$ and $\widetilde{\mathbb{M}}_X$ are isomorphic in the category of complete $\mathbb{M}$-modules.*

Let us write $\widetilde{\mathbb{M}}_\emptyset$ simply as $\widetilde{\mathbb{M}}$.

▶ **Proposition 16.** *$\widetilde{L}X \cong \mathbb{M} \times X \cup \widetilde{\mathbb{M}}$.*

Proof Sketch. The argument relies on the law of excluded middle: there are precisely two kinds of directed sequences over $\mathbb{M}_X$:

1. those, which contain an element of the from $(a, \mathsf{inl}\, x)$;
2. those, which only contain elements of the form $(a, \mathsf{inr}\,\bot)$.

In the first case, the limit is $(a, \mathsf{inl}\, x)$ (a is uniquely determined, because $(a, \mathsf{inl}\, x)$ and $(a', \mathsf{inl}\, x')$ are incompatible, unless $a = a'$ and $x = x'$). In the latter case, the limit is $(c, \mathsf{inr}\,\bot)$ where where c is the least upper bound over all such a that $(a, \mathsf{inr}\,\bot)$ is in the sequence. This produces the dichotomy of $\widetilde{L}X$ as $\mathbb{M} \times X \cup \widetilde{\mathbb{M}}$. ◀

We thus obtain agreement with Example 1. Let us again revisit Examples 2 and 4.

▶ **Example 17.** By Proposition 16, $\widetilde{L}X$ is completely determined by the initial complete $\mathbb{M}$-module $\widetilde{\mathbb{M}}$. Hence we stick to the latter. We use the fact that classically directed sequence completion and monotone sequence completion coincide. Hence w.l.o.g. we view $\widetilde{\mathbb{M}}$ as a quotient of the space of monotone sequences.

1. Completion of natural numbers yields natural numbers extended with infinity: $\widetilde{\mathbb{N}} \cong \mathbb{N} \cup \{\infty\} = \bar{\mathbb{N}}$.
2. For non-negative rational numbers we obtain $\widetilde{\mathbb{Q}}_+ \cong \mathbb{Q}_+ \uplus (\mathbb{R}_+ \smallsetminus \{0\}) \uplus \{\infty\} \cong \bar{\mathbb{Q}}_+ \uplus (\mathbb{R}_+ \smallsetminus \{0\})$. That is, a monotone sequence $s_1 \leqslant s_2 \leqslant \ldots$ is either eventually constant, i.e. has a rational number as a least upper bound, or unbounded and hence its least upper bound is infinity ∞, or, finally, the sequence is Zeno, i.e. its supremum which can be rational or irrational is not reached. For example, the sequences
$$1/2 \leqslant 3/4 \leqslant 7/8 \leqslant \ldots \qquad \text{and} \qquad 1 \leqslant 1 \leqslant 1 \leqslant \ldots$$
both tend to 1, but neither directed sequence completion nor the free object construction (which agree, as we have seen) identify them.
3. Similarly, for real numbers: $\widetilde{\mathbb{R}}^+ \cong \mathbb{R}_+ \uplus (\mathbb{R}_+ \smallsetminus \{0\}) \uplus \{\infty\} \cong \bar{\mathbb{R}}_+ \uplus (\mathbb{R}_+ \smallsetminus \{0\})$. That is, except for 0 and ∞, every real number in $\widetilde{\mathbb{R}}^+$ is counted twice: as a Zeno value and as a non-Zeno value.
4. By completing finite strings we expectedly obtain finite and infinite strings: $\widetilde{A}^\star = A^\star \cup A^\omega = A^{\leqslant\omega}$.
5. Recall the monoid $A^{[0,\mathbb{R}_+)}$ of finite trajectories **6.** from Example 2 and the monoid $A^{[0,\bar{\mathbb{R}}_+)}$ of finite and infinite trajectories **4.** from Example 4. Then, analogously to the case of real numbers, we obtain $\widetilde{A^{[0,\mathbb{R}_+)}} \cong A^{[0,\bar{\mathbb{R}}_+)} \uplus (A^{[0,\mathbb{R}_+)} \smallsetminus \{(0,!)\})$.

6 Conservatively Complete Monoid Modules

The effect of duplicating values caused by Zeno behaviour makes definite computational sense, however it might also be undesirable. The way we defined the monad $\widetilde{\mathsf{L}}$ as a result of a universal construction allows us to remedy it easily.

▶ **Definition 18** (Conservatively Complete Monoid Modules). *A complete monoid $\mathbb{M}$-module is* conservatively complete *if it satisfies the following additional axiom: for every directed sequence $(a_i)_i$ in $\mathbb{M}$, such that the least upper bound $\bigvee_i a_i$ exists, the directed sequence $(a_i \triangleright \bot)_i$ has the least upper bound $(\bigvee_i a_i) \triangleright \bot$.*

Again, conservatively complete monoid $\mathbb{M}$-modules form a category $\mathbf{Alg}_{\bar{\mathsf{L}}}$ under the same morphisms as complete monoid modules (i.e. $\mathbf{Alg}_{\bar{\mathsf{L}}}$ is a full subcategory of $\mathbf{Alg}_{\widetilde{\mathsf{L}}}$). By replicating the previous construction of a free object $\bar{L}X$, we obtain a monad $\bar{\mathsf{L}}$, for which a complete analogue of Theorem 7 holds.

▶ **Theorem 19.**
1. *The forgetful functor $U\colon \mathbf{Alg}_{\bar{\mathsf{L}}} \to \mathbf{Set}$ has a left adjoint $F\colon \mathbf{Set} \to \mathbf{Alg}_{\bar{\mathsf{L}}}$ inducing a monad over $\bar{L} = UF$ with unit η and Kleisli lifting $(-)^\star$.*
2. *$\bar{\mathsf{L}}$ is enriched over directed complete partial orders, and moreover Kleisli composition is strict on both sides.*
3. *$\bar{\mathsf{L}}$ is an Elgot monad with the iteration operator $(f\colon X \to \bar{L}(Y \uplus X))^\dagger$ calculated as a least fixed point of the map $[\eta,-]^\star f\colon (X \to \bar{L}Y) \to (X \to \bar{L}Y)$.*

Again, we write $\bar{\mathbb{M}}$ instead of $\bar{\mathbb{M}}_\emptyset$.

Next, we would like to establish an analogue of Theorem 14 in order to be able to explicitly calculate $\bar{L}X$. To that end, we need to rebase our approach the construction on *conservative completion* of partial orders. In order to facilitate the corresponding construction [14], until the end of this section we impose the following further assumption which is satisfied by all our examples.

▶ **Assumption 20.** *Let $(a_i)_i$ and $(b_i)_i$ be directed sequences over $\mathbb{M}$.*

1. *If $\bigvee_i b_i$ exists then $\bigvee_i a + b_i = a + \bigvee_i b_i$.*
2. *If $\bigvee_i a_i$ exists and for every i there exists j such that $a_i \leqslant b_j$ then either $\bigvee_i b_i$ exists or $\bigvee_i a_i \leqslant b_j$ for some j.*

Recall the definitions of $\mathbb{M}_X$ and $\widetilde{\mathbb{M}}_X$ from Section 5 and let $\bar{\mathbb{M}}_X$ be constructed in the same way as $\widetilde{\mathbb{M}}_X$, but with the following additional clause added to the equivalence relation $\sim$:

$$(a_i, \mathsf{inr}\,\bot)_i \sim (a, \mathsf{inr}\,\bot)_i \qquad \text{whenever} \qquad a = \bigvee\nolimits_i a_i. \tag{2}$$

Modulo this change, the remaining definition of the complete $\mathbb{M}$-module structure on $\bar{\mathbb{M}}_X$ is the same as for $\widetilde{\mathbb{M}}_X$. The following characterization is a counterpart of Theorem 14 and Proposition 16.

▶ **Theorem 21.** *In classical set theory, $((\bar{\mathbb{M}}_X, \rhd, \bot, \lesssim, \tilde{\tilde{\bigvee}}), \eta)$ is the free conservatively complete $\mathbb{M}$-module on X with $\eta(x) = [(0, \mathsf{inl}\, x)]_i$, and $\bar{\mathbb{M}}_X \cong \mathbb{M} \times X \cup \bar{\mathbb{M}}$.*

Proof. The fact that under the above strengthening of $\sim$ the definition of the complete $\mathbb{M}$-module structure of $\bar{\mathbb{M}}_X$ remains valid relies on Assumption 20. For example, we need to make sure that $(a \rhd_X (b_i, \mathsf{inr}\,\bot))_i \sim (a \rhd_X (b, \mathsf{inr}\,\bot))_i$ for all $a, b_i \colon \mathbb{M}$ whenever $b = \bigvee_i b_i$. Equivalently, we need to show $(a + b_i, \mathsf{inr}\,\bot)_i \sim (a + b, \mathsf{inr}\,\bot)_i$, which follows from Assumption 20 (1). Similarly, correctness of $\tilde{\tilde{\bigvee}}$ requires Assumption 20 (2). ◀

▶ **Example 22.** Let us revisit some previous examples, again, assuming that **Set** is a classical set theory. Analogously to the case of $\widetilde{\mathbf{L}}$, for $\bar{\mathbf{L}}$ we again obtain the property that $\bar{L}X$ is isomorphic to $\mathbb{M} \times X \cup \bar{\mathbb{M}}$, hence it suffices to consider $\bar{\mathbb{M}}$.

Note that the new law (2) is often ineffective, hence e.g. $\widetilde{\mathbb{N}} = \bar{\mathbb{N}}$, and $\widetilde{A}^\star = \overline{A^\star}$. However, as expected, $\overline{\mathbb{R}_+}$ consists precisely of extended reals, i.e. $\overline{\mathbb{R}_+} = \mathbb{R}_+ \cup \{\infty\}$. Analogously, $\overline{\mathbb{Q}_+} = \overline{\mathbb{R}_+}$. Finally, for the monoid of trajectories, $\overline{A^{[0,\mathbb{R}_+)}} \cong A^{[0,\bar{\mathbb{R}}_+)}$.

7 Formalization in HoTT/Cubical Agda

We next embark on the details of our formalization of the material of Sections 4 and 6 using the means of homotopy type theory (HoTT). The latter is an extension of intensional Martin-Löf type theory (MLTT) obtained by interpreting types A as topological spaces, inhabitants of types $a \colon A$ as the corresponding points, and identity types $\mathsf{Id}_A(a, b)$ as spaces of continuous paths from $a \colon A$ to $b \colon A$ within A, subject to homotopy equivalence. We use the standard Agda notation $a \;\equiv\; b$ for the identity type $\mathsf{Id}_A(a, b)$ from now on.

Among various benefits and far reaching implications of HoTT, the critical feature we need here are higher inductive-inductive types (HIITs), which in particular enable construction of free objects in the style of category theory. We carry out our formalization in the recently emerged cubical extension [22] of the Agda proof assistant – while Agda is generally based on MLTT, the cubical extension adds full support of HoTT (in the form of *cubical type theory* [7]). As a result, cubical Agda provides a rather accurate way for designing machine checked proofs in the style of HoTT, and here we dim the distinction between HoTT and cubical Agda as much as possible. We note that we only involve particular HIITs, called *quotient inductive-inductive types* (QIITs) [3], which are a special case of HIITs – this specialization is completely explicit in cubical Agda, i.e. HIITs are available via native language primitives while QIITs are expressible as certain HIITs.

In our formalization we closely follow the previous work on constructing the partiality monad (which is in our setting the duration monad over the trivial monoid) and the subsequent formalization in cubical Agda by Danielsson [8].

Since Agda supports the *propositions-as-types* discipline, types can be read as propositions and the corresponding terms as proofs. Hence, universal $\forall$ and existential $\exists$ quantifiers have the same meaning as dependent product Π and dependent sum Σ operators correspondingly. This is a standard convention for Agda, which we apply to improve readability (for technical reasons we use slightly unusual syntax for existential quantification: $\exists[\ x\ ]\ \phi$ instead of $\exists x\ \phi$). Moreover, for the same purpose, we use the disjunction symbol $\vee$ for coproducts $\uplus$ and the conjunction symbol $\wedge$ for products $\times$. For example, the following self-explanatory Agda code

```
IsProp A = ∀ (x y : A) → x ≡ y
IsSet A = ∀ (x y : A) → IsProp (x ≡ y)
IsDec A = A ∨ ¬ A
```

defines correspondingly (mere) propositions, sets and decidable types.

A derivable facility of HoTT is the *propositional truncation* operator ‖_‖ sending any type A to the type $\|A\|$ obtained by quotienting A under the equality $x \equiv y$ for all $x\ y : A$, which is implemented as follows:

```
data ‖_‖ (A : Set ℓ) : Set ℓ where
  |_| : A → ‖ A ‖
  ‖‖-prop : IsProp ‖ A ‖
```

This provides a simple example of a quotient inductive type (QIT), i.e. an inductively defined set with constructors for equalities. Such types already generally go beyond MLTT. Next, given an infinite sequence $\sigma \colon \mathbb{N} \to A$ over a partially ordered set A, the following definitions

```
Inc σ = ∀ (n : ℕ) → σ n ⩽ σ (suc n)
Dir σ = ∀ (n m : ℕ) → ∃[ k ] (σ n ⩽ σ k ∧ σ m ⩽ σ k)
‖Dir‖ σ = ∀ (n m : ℕ) → ‖ ∃[ k ] (σ n ⩽ σ k ∧ σ m ⩽ σ k) ‖
```

identify *monotone* (increasing), *intensionally directed* and *extensionally directed* sequences correspondingly. The intensional version of directedness for any two numbers n and m produces a number k with an obvious property. The extensional version ensures that such a number exists, without producing it. Observe that Inc, Dir and ‖Dir‖ are arranged by strength: if σ is monotone, then it is intensionally directed (k is the maximum of n and m), and if σ is intensionally directed then it is extensionally directed (by using |_| to forget the choice of k). Furthermore, observe that Inc, Dir and ‖Dir‖ induce the corresponding notions of Inc-complete, Dir-complete and ‖Dir‖-complete partial orders, i.e. those partial orders in which all least upper bounds of the corresponding sequences exist. These notions are therefore arranged in the opposite direction: ‖Dir‖-completeness implies Dir-completeness, and the latter implies Inc-completeness.

The discrepancy between Dir and ‖Dir‖ can be clarified in terms of the axiom of countable choice, which can be expressed e.g. as follows:

```
ACω {ℓ} = ∀ (P : ℕ → Set ℓ) → (∀ n → ‖ P n ‖) → ‖ (∀ n → P n) ‖
```

This can be read as the statement that any proof of inhabitance of $P\ n$ for every n, can be converted into a proof of existence of a corresponding choice function. Intuitively, under ACω one should be able to convert an extensionally directed sequence to an intensionally directed one by successively pushing the truncation operator ‖_‖ upwards and then applying the elimination principle for ‖_‖. This indeed works, and in summary we have the following set of results:

▶ **Proposition 23.** *Let (a), (b) and (c) stand for completeness of a fixed set A w.r.t. ‖Dir‖, Dir and Inc correspondingly. Then*

- *(a) ⇒ (b) ⇒ (c);*
- *(b) ⇒ (a) under countable choice;*
- *(c) ⇒ (a) under the decidability of $\leqslant$ on A (i.e. under $\forall\ (x\ y\ :\ A)\ \to$ IsDec $(x \leqslant y)$).*

Proof Sketch. Consider the last clause, which is the one we did not discuss yet. The idea is based on Exercise 3.19. from the HoTT book [20], which can be formalized as follows:

```
restore-ωC : ∀ (P : ℕ → Set ℓ) → (∀ (n : ℕ) → IsDec (P n)) → ‖ ∃[ n ] P n ‖ → ∃[ n ] P n
```

That is, under decidability of all $P\,n$, the fact that there exists n satisfying P implies a constructive procedure for producing such an n. In our implementation such a procedure simply finds the first n that satisfies P, and that critically depends on the decidability assumption – otherwise the very concept "first n satisfying P" cannot be realized. Using restore-ωC, and the decidability assumption for $\leqslant$, it is easy to select a monotone subsequence from any extensionally directed sequence and show coincidence of the corresponding least upper bounds. ◀

For decidable $\leqslant$, ‖Dir‖-, Dir- and Inc-completeness are therefore equivalent, which is the case of the partiality monad. While the partiality monad is based on Inc-completeness, our implementation of $\tilde{\mathbf{L}}$ and $\bar{\mathbf{L}}$ is based on Dir-completeness, as we explain next.

For $\tilde{\mathbf{L}}$ we introduce an HIIT of a mutually dependent carrier (implicitly parametrized by an argument A of type Set $(\ell \sqcup \ell')$) and a binary relation $\sqsubseteq$ on it, w.r.t. an ordered monoid $\mathbb{M}$ with a carrier from Set ℓ and a partial order relation on $\mathbb{M}$ from Set ℓ':

```
data L̃ : Set (ℓ ⊔ ℓ′)
data _⊑_ : L̃ → L̃ → Set (ℓ ⊔ ℓ′)
```

The following forward reference asserts that $\tilde{L}$ will be a partial order

```
PO-⊑ : PartialOrder L̃
```

Then we introduce the constructors for $\tilde{L}$:

```
data L̃ where
  _▷_ : 𝕄 → L̃ → L̃
  ⊥ : L̃
  ⨆ : DirSeq PO-⊑ → L̃
  η : A → L̃
  ⊑-antisym : ∀ (x y : L̃) → x ⊑ y → y ⊑ x → x ≡ y
```

where DirSeq PO-⊑ is the type of intensionally directed sequences over $\tilde{L}$. The corresponding definition of _⊑_ is more technical, and we omit it here. This definition contains all the necessary axioms and in particular allows us to define PO-⊑. Moreover, it is asserted that IsProp $(x \sqsubseteq y)$ is inhabited, which implies that our carrier is a set (Theorem 7.2.2 of the HoTT book), i.e. the HIIT we define is indeed a QIIT. The most technically involved remaining part of the construction is the definition of the elimination principle together with the proof that it implies initiality of $\tilde{L}$ which in the categorical sense is the same as freeness of each $\tilde{L}X$ as an $\mathbb{M}$-module on X. The construction of $\bar{L}$ is analogous – essentially we add the conservative completeness property as a new axiom to the definition of _⊑_.

Our definitions of $\tilde{L}$ and $\bar{L}$ are based on the intensional notion of directedness. It is currently not clear to us if those can also be based on the corresponding extensional notion, and whether this would bring any benefits over the present formalization. We regard this as an issue for further work. For another alternative, it would be perfectly possible to base $\widetilde{L}$ and $\bar{L}$ on Inc-completeness. We avoided that for a strategic reason: although the results of Section 5 currently hold under rather strong classicality assumptions, some of them must hold under weaker assumptions such as the axiom of countable choice. The challenges we would face then would be similar to those one faces when proving completeness of Cauchy reals. That standardly relies on a diagonalization argument roughly stating that a directed sequence of directed sequences can be converted to a directed sequence. This argument is constructively valid for Dir-completeness, but for Inc-completeness it seems to indispensably rely on decidability of inequality of the underlying set.

8 Conclusions and Further Work

We proposed a constructive formalization of hybrid semantics by combining ideas from category theory, type theory and domain theory and justified the results by an implementation in the Agda proof assistant extended with support of cubical type theory [7]. On the one hand, we closely followed the previous work on constructing the partiality monad, and on the other hand, complemented this construction with an explicit "time" dimension in the form of an ordered monoid. We thus reinforce the importance of quotient inductive-inductive types (QIITs) [3] which have previously been used for defining constructive counterparts of important semantic notions such as Cauchy reals and non-terminating computations so that even the principle of countable choice is avoided. Our analysis also identifies the importance of the notion of conservative completion [23], which seems to be little known in the computer science community, possibly because this notion only becomes relevant when dealing with non-discrete data types, once Zeno effects come into play. In contrast to the partiality monad case, our characterization in terms of directed sequence completion is only established under strong classicality assumptions and not with countable choice as the only additional axiom. We leave it as an important pending question for further work to check if our results to this effect can be improved. In a nutshell, a potential positive answer would amount to implementing the relevant parts of Section 5 in cubical Agda with the axiom of countable choice postulated.

Our work is motivated by a simple deterministic hybrid language HybCore for hybrid computation [10], which we now provided with a constructive semantics. As a next step, we are planning to extend HybCore with further features such as concurrency and non-determinism in a principled fashion. In fact, the majority of existing work on hybrid systems intertwine hybridness and non-determinism (even though, conceptually, these are independent computational effects). From a type-theoretic perspective, an interesting insight into relating partiality and nondeterminism is provided by Veltri [21], who proposed to build a constructive analogue of the countable powerset monad as a quotient inductive type (instead of a QIIT!), from which the partiality monad is then ingeniously carved out. The technical benefit of this approach is that it enables construction of the partiality monad in environments which do not support QIITs. We consider this approach more broadly as a way to relate partiality and non-determinism, and in particular, we plan to investigate its potential for constructing non-deterministic hybrid monads.

References

1 Samson Abramsky and Achim Jung. Domain theory. In *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Oxford University Press, 1994.

2 Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and concrete categories*. John Wiley & Sons Inc., New York, 1990.

3 Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient inductive-inductive types. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures*, pages 293–310, Cham, 2018. Springer International Publishing.

4 Thorsten Altenkirch, Nils Danielsson, and Nicolai Kraus. Partiality, revisited - the partiality monad as a quotient inductive-inductive type. In Javier Esparza and Andrzej Murawski, editors, *Foundations of Software Science and Computation Structures, FOSSACS 2017*, volume 10203 of *LNCS*, pages 534–549, 2017.

5 Abhishek Anand and Ross Knepper. ROSCoq: Robots powered by constructive reals. In Christian Urban and Xingyuan Zhang, editors, *Interactive Theorem Proving*, pages 34–50, Cham, 2015. Springer International Publishing.

6 James Chapman, Tarmo Uustalu, and Niccolò Veltri. Quotienting the delay monad by weak bisimilarity. In *Theoretical Aspects of Computing, ICTAC 2015*, volume 9399 of *LNCS*, pages 110–125. Springer, 2015.

7 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

8 Nils Anders Danielsson. Code related to the paper "Partiality, Revisited: The Partiality Monad as a Quotient Inductive-Inductive Type". http://www.cse.chalmers.se/~nad/listings/partiality-monad/. Accessed: 2020-02-12.

9 Sergey Goncharov, Julian Jakob, and Renato Neves. A semantics for hybrid iteration. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory (CONCUR 2018)*, LNCS. Springer, 2018.

10 Sergey Goncharov and Renato Neves. An adequate while-language for hybrid computation. In Ekaterina Komendantskaya, editor, *Proceedings of the 21th International Symposium on Principles and Practice of Declarative Programming, (PPDP 2019)*. ACM, 2019.

11 Sergey Goncharov, Lutz Schröder, Christoph Rauch, and Julian Jakob. Unguarded recursion on coinductive resumptions. *Logical Methods in Computer Science*, 14(3), 2018.

12 Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Inf. & Comp*, 185:2003, 2002.

13 Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

14 Georg Markowsky. Chain-complete posets and directed sets with applications. *Algebra universalis*, 6:53–68, 1976.

15 John C. Mitchell. *Foundations of Programming Languages*. MIT Press, Cambridge, MA, USA, 1996.

16 Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93:55–92, 1991.

17 G. Plotkin. Post graduate lecture notes on advanced domain theory (incorporating the “Pisa notes”). Dept. of Computer Science, Univ. Edinburgh, 1981.

18 J. Reynolds. *Theories of Programming Languages*. Cambridge University Press, 1998.

19 Benjamin Sherman, Luke Sciarappa, Adam Chlipala, and Michael Carbin. Computable decision making on the reals and other spaces: Via partiality and nondeterminism. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS ’18, pages 859–868, New York, NY, USA, 2018. ACM.

20 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

21 Niccoló Veltri. *A Type-Theoretical Study of Nontermination*. PhD thesis, Tallinn University of Technology, 2017.

22 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *Proceedings of the ACM on Programming Languages*, 3(ICFP):87, 2019.

23 Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1992.

24 G. Winskel. *The Formal Semantics of Programming Languages*. MIT Press, Cambridge, Massachusetts, 1993.

A Omitted Proofs

A.1 Proof of Theorem 14

We fix a complete $\mathbb{M}$-module $(\mathbb{E}, \triangleright, \bot, \sqsubseteq, \bigsqcup)$ together with a map $f\colon X \to \mathbb{E}$. Our goal is to construct a unique complete $\mathbb{M}$-module morphism $f^\star\colon \widetilde{\mathbb{M}}_X \to \mathbb{E}$ satisfying $f^\star(\eta(x)) = f(x)$. To this purpose we define an auxiliary function $\ell\colon X \uplus \{\bot\} \to \mathbb{E}$ as follows:

$$\ell(\mathsf{inl}\, x) = f(x), \qquad \ell(\mathsf{inr}\, \bot) = \bot.$$

Now, we define $f^\star\colon \widetilde{\mathbb{M}}_X \to \mathbb{E}$ as follows:

$$f^\star[(a_i, x_i)]_i = \bigsqcup\nolimits_i a_i \triangleright \ell(x_i).$$

This definition is only valid if $(a_i \triangleright \ell(x_i))_i$ is directed and $\bigsqcup_i a_i \triangleright \ell(x_i)$ does not depend on the specific representatives $(a_i, x_i)_i$ of the corresponding equivalence classes. Both these properties require us to prove that $(a, x) \sqsubseteq_X (b, y)$ implies $a \triangleright \ell(x) \sqsubseteq b \triangleright \ell(y)$, which is shown as follows:

- if $(a, \mathsf{inl}\, x) \sqsubseteq_X (b, \mathsf{inl}\, y)$ then $(a, \mathsf{inl}\, x) = (b, \mathsf{inl}\, y)$ and thus $a \triangleright \ell(\mathsf{inl}\, x) \sqsubseteq b \triangleright \ell(\mathsf{inl}\, y)$;
- if $(a, \mathsf{inr}\, \bot) \sqsubseteq_X (b, \mathsf{inl}\, y)$ then $a \leqslant b$ and thus $a \triangleright \ell(\mathsf{inr}\, \bot) = a \triangleright \bot \sqsubseteq b \triangleright \bot \sqsubseteq b \triangleright f(y) = b \triangleright \ell(\mathsf{inl}\, y)$ by the *restricted left monotonicity*, *least element* and *right monotonicity* axioms;
- if $(a, \mathsf{inr}\, \bot) \sqsubseteq_X (b, \mathsf{inr}\, \bot)$ then $a \leqslant b$ and thus $a \triangleright \ell(\mathsf{inr}\, \bot) = a \triangleright \bot \sqsubseteq b \triangleright \bot = b \triangleright \ell(\mathsf{inr}\, \bot)$ by *restricted left monotonicity.*

Next, let us show that $f^\star$ is a complete $\mathbb{M}$-module morphism, by verifying the corresponding preservation properties:

- $f^\star(a \triangleright [(b_i, x_i)]_i) = a \triangleright f^\star[(b_i, x_i)]_i$: the requisite calculation runs as follows:

$$\begin{aligned}
f^\star(a \triangleright [(b_i, x_i)]_i) &= f^\star[a \triangleright_X (b_i, x_i)]_i && \textit{(definition of } \triangleright\textit{)}\\
&= f^\star[(a + b_i,\, x_i)]_i && \textit{(definition of } \triangleright_X\textit{)}\\
&= \bigsqcup\nolimits_i (a + b_i) \triangleright \ell(x_i) && \textit{(definition of } f^\star\textit{)}\\
&= \bigsqcup\nolimits_i a \triangleright (b_i \triangleright \ell(x_i)) && \textit{(monoid action)}\\
&= a \triangleright \bigsqcup\nolimits_i b_i \triangleright \ell(x_i) && \textit{(right continuity of } \triangleright\textit{)}\\
&= a \triangleright f^\star[(b_i, x_i)]_i. && \textit{(definition of } f^\star\textit{)}
\end{aligned}$$

- $f^\star(\bot) = \bot$: $f^\star(\bot) = f^\star[(0, \mathsf{inr}\, \bot)]_i = \bigsqcup_i 0 \triangleright \ell(\mathsf{inr}\, \bot) = \bigsqcup_i 0 \triangleright \bot = \bigsqcup_i \bot = \bot$ using the definition of monoidal action.

- $[(a_i, x_i)]_i \lessapprox [(b_i, y_i)]_i$ implies $f^\star[(a_i, x_i)]_i \sqsubseteq f^\star[(b_i, y_i)]_i$: If $[(a_i, x_i)]_i \lessapprox [(b_i, y_i)]_i$ then $(a_i, x_i)_i \lesssim (b_i, y_i)_i$ meaning that for every $i\colon \mathbb{N}$ there exists a $j\colon \mathbb{N}$ such that $(a_i, x_i) \sqsubseteq_X (b_j, y_j)$. This implies $a_i \triangleright \ell(x_i) \sqsubseteq b_j \triangleright \ell(y_j) \sqsubseteq \bigsqcup_i b_i \triangleright \ell(y_i)$ for every $i\colon \mathbb{N}$ by *upper bound* and thus $f^\star[(a_i, x_i)]_i = \bigsqcup_i a_i \triangleright \ell(x_i) \sqsubseteq \bigsqcup_i b_i \triangleright \ell(y_i) = f^\star[(b_i, y_i)]_i$ by *least upper bound.*
- $f^\star\left(\overset{\approx}{\bigvee}_i[(a_{i,j}, x_{i,j})]_j\right) = \bigsqcup_i f^\star[(a_{i,j}, x_{i,j})]_j$: This is obtained as follows.

$$\begin{aligned} f^\star\left(\overset{\approx}{\bigvee}_i[(a_{i,j}, x_{i,j})]_j\right) &= f^\star[(a_{\pi_1^{-1}(i),\pi_2^{-1}(i)}, x_{\pi_1^{-1}(i),\pi_2^{-1}(i)})]_i \\ &= \bigsqcup_i a_{\pi_1^{-1}(i),\pi_2^{-1}(i)} \triangleright \ell(x_{\pi_1^{-1}(i),\pi_2^{-1}(i)}) \\ &= \bigsqcup_i \bigsqcup_j a_{i,j} \triangleright \ell(x_{i,j}) \qquad (*) \\ &= \bigsqcup_i f^\star[(a_{i,j}, x_{i,j})]_j. \end{aligned}$$

The only step that does not follow by definition is $(*)$, and we show it by *antisymmetry* of $\sqsubseteq$ as follows.

  - $(\sqsubseteq)$: Let us fix $i\colon \mathbb{N}$ and let $k_1 = \pi_1^{-1}(i)$, $k_2 = \pi_2^{-1}(i)$, and then

$$\begin{aligned} a_{\pi_1^{-1}(i),\pi_2^{-1}(i)} \triangleright \ell(x_{\pi_1^{-1}(i),\pi_2^{-1}(i)}) &= a_{k_1,k_2} \triangleright \ell(x_{k_1,k_2}) \\ &\sqsubseteq \bigsqcup_j a_{k_1,j} \triangleright \ell(x_{k_1,j}) \\ &\sqsubseteq \bigsqcup_i \bigsqcup_j a_{i,j} \triangleright \ell(x_{i,j}). \end{aligned}$$

    Since i is arbitrary, by the *least upper bound* property, $\bigsqcup_i a_{\pi_1^{-1}(i),\pi_2^{-1}(i)} \triangleright \ell(x_{\pi_1^{-1}(i),\pi_2^{-1}(i)}) \sqsubseteq \bigsqcup_i \bigsqcup_j a_{i,j} \triangleright \ell(x_{i,j})$.
  - $(\sqsupseteq)$: For all $i, j\colon \mathbb{N}$, $a_{i,j} \triangleright \ell(x_{i,j}) = a_{\pi^{-1}(\pi(i,j))} \triangleright \ell(x_{\pi^{-1}(\pi(i,j))}) \sqsubseteq \bigsqcup_i a_{\pi^{-1}(i)} \triangleright \ell(x_{\pi^{-1}(i)})$ and hence $\bigsqcup_i \bigsqcup_j a_{i,j} \triangleright \ell(x_{i,j}) \sqsubseteq \bigsqcup_i a_{\pi_1^{-1}(i),\pi_2^{-1}(i)} \triangleright \ell(x_{\pi_1^{-1}(i),\pi_2^{-1}(i)})$ by the *least upper bound* property.

Next we show commutativity of (1), i.e. that $f^\star(\eta(x)) = f(x)$, as follows: $f^\star(\eta(x)) = f^\star[(0, \mathsf{inl}\, x)]_i = \bigsqcup_i 0 \triangleright \ell(\mathsf{inl}\, x) = \bigsqcup_i 0 \triangleright f(x) = \bigsqcup_i f(x) = f(x)$ using the definition of monoidal action.

Finally, we show that the constructed morphism $f^\star$ is unique. That is, given another complete $\mathbb{M}$-module morphism $g\colon \widetilde{\mathbb{M}}_X \to \mathbb{E}$ satisfying $g(\eta(x)) = f(x)$, we show that $f^\star$ and g are equal. First, let $h\colon X \uplus \{\bot\} \to \widetilde{\mathbb{M}}_X$ be defined as $h(x) = [(0, x)]_i$. Note that $g(h(x)) = \ell(x)$ for any $x\colon X \uplus \{\bot\}$:

$$\begin{aligned} g(h(\mathsf{inl}\, x)) &= g[(0, \mathsf{inl}\, x)]_j = g(\eta(x)) = f(x) = \ell(\mathsf{inl}\, x), \\ g(h(\mathsf{inr}\, \bot)) &= g[(0, \mathsf{inr}\, \bot)]_j = g(\bot) = \bot = \ell(\mathsf{inr}\, \bot). \end{aligned}$$

The desired equation $f^\star = g$ is obtained as follows:

$$\begin{aligned} g[(a_i, x_i)]_i &= g\Big(\overset{\approx}{\bigvee}_i a_i \triangleright h(x_i)\Big) = \bigsqcup_i g(a_i \triangleright h(x_i)) = \bigsqcup_i a_i \triangleright g(h(x_i)) \\ &= \bigsqcup_i a_i \triangleright \ell(x_i) = f^\star[(a_i, x_i)]_i. \end{aligned}$$

Here, only the first step is not by definition. Let us show that, in fact, $[(a_i, x_i)]_i = \overset{\approx}{\bigvee}_i a_i \triangleright h(x_i)$. Note that $(a_i, x_i)_i \sim (a_{\pi_1^{-1}(i)}, x_{\pi_1^{-1}(i)})_i$ basically because by definition $\pi_1^{-1}(i)$ can be made arbitrary large by choosing a suitable i. Therefore $[(a_i, x_i)]_i = [(a_{\pi_1^{-1}(i)}, x_{\pi_1^{-1}(i)})]_i$ and we have

$$[(a_i, x_i)]_i = [(a_{\pi_1^{-1}(i)}, x_{\pi_1^{-1}(i)})]_i = \overset{\approx}{\bigvee}_i [(a_i, x_i)]_j = \overset{\approx}{\bigvee}_i a_i \triangleright [(0, x_i)]_j = \overset{\approx}{\bigvee}_i a_i \triangleright h(x_i),$$

which completes the proof. ◀