A Complete Finitary Refinement Type System for Scott-Open Properties

Colin Riba
ENS de Lyon, CNRS, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France

Adam Donadille
ENS de Lyon, CNRS, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France

Abstract

We are interested in proving input-output properties of functions that handle infinite data such as streams or non-wellfounded trees. We provide a finitary refinement type system which is (sound and) complete for Scott-open properties defined in a fixpoint-like logic. Working on top of Abramsky's Domain Theory in Logical Form, we build from the well-known fact that the Scott domains interpreting recursive types are spectral spaces. The usual symmetry between Scott-open and compact-saturated sets is reflected in logical polarities: positive formulae allow for least fixpoints and define Scott-open sets, while negative formulae allow for greatest fixpoints and define compact-saturated sets. A realizability implication with the expected (contra)variance on polarities allows for non-trivial input-output properties to be formulated as positive formulae on function types.

2012 ACM Subject Classification Theory of computation → Denotational semantics; Theory of computation → Logic and verification

Keywords and phrases Domain Theory, Temporal Logic, Refinement Types

Digital Object Identifier 10.4230/LIPIcs.FSCD.2026.28

Related Version *Full Version*: `https://arxiv.org/abs/2601.23082` [31]

Funding This work was partially supported by the French ANR PRCI project *ATQUA*, ANR-25-CE48-1428-01.

1 Introduction

We are interested in input-output specifications of functions that handle infinite data, such as streams or non-wellfounded trees. Consider the function $\mathsf{count} : \mathsf{Stream\,Bool} \to \mathsf{Stream\,Nat}$ defined as $\mathsf{count} = \mathsf{countrec}\ \mathsf{0}$, where

$$\begin{aligned} \mathsf{countrec} :&\ \ \mathsf{Nat} \longrightarrow \mathsf{Stream\,Bool} \longrightarrow \mathsf{Stream\,Nat} \\ \mathsf{countrec}\ n\ (b :: x) =&\ \ \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{countrec}\ (\mathsf{1}{+}n)\ x)\ \mathsf{else}\ n :: (\mathsf{countrec}\ n\ x) \end{aligned}$$

Given a stream of Booleans x, the stream of natural numbers $(\mathsf{count}\ x)$ retains, for each occurrence of ff in x, the number of tt's seen so far in x. It is clear that if there are infinitely many tt's and infinitely many ff's in x, then $(\mathsf{count}\ x)$ contains arbitrary large numbers.

Working with a simply-typed λ-calculus with sums and recursive types (a.k.a. FPC) [30], we are interested in specifications interpreted in a denotational semantics: since a stream (as opposed to e.g. an integer) is an inherently infinite object, a specification for (e.g.) count should hold for any stream whatsoever, and not only for those definable in a given syntax.

This leads us to consider properties on infinite datatypes in Scott domains. Logics on top of domains are known since quite a long time. We resort to Abramsky's paradigm of "Domain Theory in Logical Form" [1, 2], which allows one to systematically generate a logic from a domain representing a type. These logics are obtained by Stone duality, the core of a rich interplay between domain theory, logic and topology (cf e.g. [20, 39, 42, 4, 10, 40, 13, 12]).

In a previous work [32] (based on [7]), we considered a refinement type system which is sound and complete for saturated (i.e. upward-closed) properties on Scott domains. This strictly extends [1, 2]. Indeed, saturated properties cover least and greatest fixpoints of negation-free formulae in modal μ-calculi on the corresponding types. On streams, this includes usual modalities of Linear Temporal Logic (LTL), such as $\Diamond$ (*eventually*, i.e. "there is a position such that") and $\Box$ (*always*, i.e. "for all positions") [33]. Logics like LTL, CTL (Computation Tree Logic) or modal μ-calculi are widely used to formulate properties on infinite objects (see e.g. [5, 14, 8]).

On the other hand, only compact-open properties are available in [1, 2]. But for each compact-open set K of streams, there is some $n \in \mathbb{N}$ such that membership to K is completely determined by prefixes of length n. This excludes proper uses of $\Diamond$ and $\Box$. However, the systems in [7, 32] are *infinitary*, in the sense that some rules have infinitely many premises.

In this paper, we devise a *finitary* refinement type system which is sound and complete for a family of Scott-open properties allowing for specifications of infinitary behaviours. This extends [1, 2]. Types are refined by a (negation-free) fixpoint-like logic which relies on the well-known fact that the Scott domains interpreting recursive types are *spectral spaces*.

In a spectral space (and more generally in a *stably compact space* [24]), there is a symmetry between the open sets and the compact-saturated ones. We reflect this symmetry in the logic: we have *positive* formulae which are closed under least fixpoints and define Scott-open sets, and *negative* formulae which are closed under greatest fixpoints and define compact-saturated sets. Input-output specifications of functions are written using a realizability implication $\Vdash\!\!\rightarrow$ with the expected (contra)variance w.r.t. polarities.

On streams, positive formulae are closed under $\Diamond$, while negative formulae are closed under $\Box$. Given a positive formula φ and a negative formula ψ, we have a positive formula $\Box\psi \Vdash\!\!\rightarrow \Diamond\varphi$ which selects those functions that take a stream whose elements all satisfy ψ to a stream with at least one element satisfying φ. Consider now the composite

$$\mathsf{count} \circ \mathsf{bft} : \ \mathsf{Tree\,Bool} \longrightarrow \mathsf{Stream\,Nat}$$

where $\mathsf{bft}\colon \mathsf{Tree}\,\sigma \to \mathsf{Stream}\,\sigma$ is a breadth-first tree traversal. Negative formulae on trees are closed under the usual CTL-like modalities $\forall\Box$ (*invariantly*, i.e. "for all nodes") and $\exists\Box$ (*potentially always*, i.e. "there is an infinite path on which for all nodes"). Thanks to the realizability implication, for each $n \in \mathbb{N}$ there is a positive formula which expresses the following specification for $\mathsf{count} \circ \mathsf{bft}$: Given a totally defined tree of Booleans, if the children of the root have an infinite path with only tt's and an infinite path with only ff's, then $\mathsf{count} \circ \mathsf{bft}$ produces a stream which contains a totally defined number $\geq n$.

We prove a *Positive Completeness* result: our type system derives all sound positive specifications of λ-terms (thus including the above positive specifications for $\mathsf{count} \circ \mathsf{bft}$).

Positive completeness implies that checking positive formulae on λ-terms is semi-decidable. This is in line with the intuition that membership to a Scott-open set is akin to a reachability problem: evaluate the (semantics of the) λ-term until knowing that the open set is met. Our refinement type system essentially amounts to a syntactic approach to this reachability problem. We also note that checking positive formulae on λ-terms is *not* decidable.

Besides, an adaptation of Higher-Order Model Checking to input-output specifications for a specific format of higher-order tree transducers (weaker than our FPC) has been shown to be undecidable in [22].

At the technical level, our logic allows for prenex normal forms. To this end, least and greatest fixpoints are decomposed as finite iterations indexed by variables which are existentially and universally quantified, respectively. This builds on ideas in [19] (which is based on guarded recursion and states no completeness result).

$$\frac{}{\mathcal{E} \vdash \langle\rangle : \mathsf{Unit}} \qquad \frac{\mathcal{E}, x : \tau \vdash M : \tau}{\mathcal{E} \vdash \mathsf{fix}\, x.M : \tau} \qquad \frac{\mathcal{E} \vdash M : \tau[\mathsf{rec}\,\alpha.\tau/\alpha]}{\mathcal{E} \vdash \mathsf{fold}(M) : \mathsf{rec}\,\alpha.\tau} \qquad \frac{\mathcal{E} \vdash M : \mathsf{rec}\,\alpha.\tau}{\mathcal{E} \vdash \mathsf{unfold}(M) : \tau[\mathsf{rec}\,\alpha.\tau/\alpha]}$$

$$\frac{\mathcal{E} \vdash M : \tau_i}{\mathcal{E} \vdash \mathsf{in}_i(M) : \tau_1 + \tau_2} \qquad \frac{\mathcal{E} \vdash M : \tau_1 + \tau_2 \qquad \mathcal{E}, x_1 : \tau_1 \vdash N_1 : \sigma \qquad \mathcal{E}, x_2 : \tau_2 \vdash N_2 : \sigma}{\mathcal{E} \vdash \mathsf{case}\ M\ [x_1 \mapsto N_1 \mid x_2 \mapsto N_2] : \sigma}$$

Figure 1 Typing rules for pure types, where $i \in \{1, 2\}$.

Organisation of the paper. Our simply-typed λ-calculus is discussed in § 2. The logic is introduced in § 3, while refinement types are presented in § 4. Section 5 covers denotational semantics, soundness and positive completeness, as well as some necessary material on spectral spaces. Finally, the conclusion (§ 6) discusses some further works and compare with more related works. Proofs are available in the Appendices of the full version [31].

2 The Pure System

The *pure types* (notation $\tau, \sigma, \dots$) are the closed types over the grammar

$$\tau \ ::= \ \mathsf{Unit} \mid \tau \times \tau \mid \tau \to \tau \mid \alpha \mid \mathsf{rec}\,\alpha.\tau \mid \tau + \tau$$

where α ranges over an infinite supply of *type variables*, and where $\mathsf{rec}\,\alpha.\tau$ binds α in τ.

We consider λ-terms from the grammar

$$M, N \ ::= \ \langle\rangle \mid \langle M, N\rangle \mid \pi_1(M) \mid \pi_2(M) \mid x \mid \lambda x.M \mid MN \mid \mathsf{fix}\, x.M \\ \mid \mathsf{fold}(M) \mid \mathsf{unfold}(M) \mid \mathsf{in}_1(M) \mid \mathsf{in}_2(M) \mid \mathsf{case}\ M\ [x_1 \mapsto N_1 \mid x_2 \mapsto N_2]$$

The term formers $\mathsf{fold}, \mathsf{unfold}, \pi_1, \pi_2, \mathsf{in}_1, \mathsf{in}_2$ are often written in curried form, so that e.g. $(\mathsf{fold}\, M)$ stands for $\mathsf{fold}(M)$. We write $M \circ N$ for $\lambda x.M(N\, x)$.

Terms are typed with judgements of the form $\mathcal{E} \vdash M : \tau$, where the *typing context* $\mathcal{E}$ is a list $x_1 : \sigma_1, \dots, x_n : \sigma_n$ with $x_i \neq x_j$ if $i \neq j$. The typing rules are those in Figure 1, together with the *basic rules* in Figure 7a *restricted to pure types* (i.e. with $\mathcal{E}$ as above and T, U pure types). Of course, each type τ is inhabited by the "diverging" term $\Omega_\tau \coloneqq \mathsf{fix}\, x.x : \tau$.

▶ **Example 2.1.** The type $\mathsf{Bool} \coloneqq \mathsf{Unit} + \mathsf{Unit}$ represents Booleans, with $\mathsf{tt} \coloneqq \mathsf{in}_1(\langle\rangle)$ and $\mathsf{ff} \coloneqq \mathsf{in}_2(\langle\rangle)$. The type of natural numbers is $\mathsf{Nat} \coloneqq \mathsf{rec}\,\alpha.\,\mathsf{Unit} + \alpha$. For each $n \in \mathbb{N}$, we have a term $\mathsf{n} : \mathsf{Nat}$ defined as $\mathsf{0} \coloneqq \mathsf{fold}(\mathsf{in}_1\,\langle\rangle)$ and $\mathsf{n}+\mathsf{1} \coloneqq (\mathsf{1}+)\mathsf{n}$, where $(\mathsf{1}+) : \mathsf{Nat} \to \mathsf{Nat}$ is the successor function $\lambda x.\,\mathsf{fold}(\mathsf{in}_2\, x)$. We can represent "$+\infty$" as $\mathsf{inf} \coloneqq \mathsf{fix}\, x.\,\mathsf{1}{+}x : \mathsf{Nat}$.

The type of streams over σ is $\mathsf{Stream}\,\sigma \coloneqq \mathsf{rec}\,\alpha.\,\sigma \times \alpha$. It is equipped with the constructor $\mathsf{Cons} \coloneqq \lambda h.\lambda t.\,\mathsf{fold}\,\langle h, t\rangle : \sigma \to \mathsf{Stream}\,\sigma \to \mathsf{Stream}\,\sigma$. We use the infix notation $(M :: N)$ for $(\mathsf{Cons}\, M\, N)$. The usual *head* and *tail* functions are $\mathsf{hd} \coloneqq \lambda s.\,\pi_1(\mathsf{unfold}\, s) : \mathsf{Stream}\,\sigma \to \sigma$ and $\mathsf{tl} \coloneqq \lambda s.\,\pi_2(\mathsf{unfold}\, s) : \mathsf{Stream}\,\sigma \to \mathsf{Stream}\,\sigma$.

The type of binary trees over σ is $\mathsf{Tree}\,\sigma \coloneqq \mathsf{rec}\,\alpha.\,\sigma \times (\alpha \times \alpha)$. The constructor $\mathsf{Node} : \sigma \to \mathsf{Tree}\,\sigma \to \mathsf{Tree}\,\sigma \to \mathsf{Tree}\,\sigma$ and the destructors $\mathsf{label} : \mathsf{Tree}\,\sigma \to \sigma$ and $\mathsf{left}, \mathsf{right} : \mathsf{Tree}\,\sigma \to \mathsf{Tree}\,\sigma$ are defined similarly as Cons, hd, tl on streams. ⌟

▶ **Example 2.2.** Table 1 defines functions on streams and trees mentioned in § 1. The stream function count was discussed in § 1. The notation $\mathsf{if}\ M\ \mathsf{then}\ N_1\ \mathsf{else}\ N_2$ stands for the term $\mathsf{case}\ M\ [\_ \mapsto N_1 \mid \_ \mapsto N_2]$. On trees, the function bft implements Martin Hofmann's breadth-first traversal (see e.g. [6, 19, 32, 21]). It uses the recursive type $\mathsf{Cont}\,\sigma \coloneqq \mathsf{rec}\,\alpha.\,(\alpha \to \sigma) \to \sigma$. Note that the recursive type corresponding to $\mathsf{Cont}\,\sigma$ in [6, 19, 21] has an additional constructor (in place of our fixpoint $\mathsf{Over} : \mathsf{Cont}\,\sigma$). See Example 5.2 (§ 5.1). ⌟

Table 1 Functions on Streams and Trees.

$$\begin{array}{l}
\textbf{countrec}: \quad \mathsf{Nat} \longrightarrow \mathsf{Stream\,Bool} \longrightarrow \mathsf{Stream\,Nat} \\
\qquad := \ \mathsf{fix}\, g.\lambda n.\lambda x.\ \mathsf{if}\ (\mathsf{hd}\, x)\ \mathsf{then}\ (g\ (1{+}n)\ (\mathsf{tl}\, x))\ \mathsf{else}\ (n :: (g\ n\ (\mathsf{tl}\, x))) \\
\\
\qquad\qquad \textbf{count} \ := \ (\mathsf{countrec}\ 0): \quad \mathsf{Stream\,Bool} \longrightarrow \mathsf{Stream\,Nat}
\end{array}$$

$$\begin{array}{ll}
\textbf{extract}: \quad \mathsf{Cont}\,\sigma \longrightarrow \sigma & \textbf{Over}: \quad \mathsf{Cont}\,\sigma \\
\qquad := \ \mathsf{fix}\, e.\lambda c.\, \mathsf{unfold}\ c\ e & \qquad := \ \mathsf{fix}\, c.\, \mathsf{fold}(\lambda k.\ k\ c)
\end{array}$$

$$\begin{array}{l}
\textbf{bftrec}: \quad \mathsf{Tree}\,\sigma \longrightarrow \mathsf{Cont}(\mathsf{Stream}\,\sigma) \longrightarrow \mathsf{Cont}(\mathsf{Stream}\,\sigma) \\
\qquad := \ \mathsf{fix}\, g.\lambda t.\lambda c.\, \mathsf{fold}\,\big(\lambda k.\ (\mathsf{label}\, t) :: \big(\mathsf{unfold}\ c\ \big(k \circ (g(\mathsf{left}\, t)) \circ (g(\mathsf{right}\, t))\big)\big)\big) \\
\\
\qquad\qquad \textbf{bft} \ := \ \lambda t.\ \ \mathsf{extract}(\mathsf{bftrec}\ \ t\ \ \mathsf{Over}): \quad \mathsf{Tree}\,\sigma \longrightarrow \mathsf{Stream}\,\sigma
\end{array}$$

$$\frac{}{\mathsf{True} \in \mathcal{L}^s(\Sigma;\tau)} \qquad \frac{\varphi, \psi \in \mathcal{L}^s(\Sigma;\tau)}{\varphi \wedge \psi \in \mathcal{L}^s(\Sigma;\tau)} \qquad \frac{}{\mathsf{False} \in \mathcal{L}^s(\Sigma;\tau)} \qquad \frac{\varphi, \psi \in \mathcal{L}^s(\Sigma;\tau)}{\varphi \vee \psi \in \mathcal{L}^s(\Sigma;\tau)}$$

(a) Propositional connectives.

$$\frac{}{[\langle\rangle] \in \mathcal{L}^s(\Sigma;\mathsf{Unit})} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma;\tau_1)}{[\pi_1]\varphi \in \mathcal{L}^s(\Sigma;\tau_1 \times \tau_2)} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma;\tau_2)}{[\pi_2]\varphi \in \mathcal{L}^s(\Sigma;\tau_1 \times \tau_2)}$$

$$\frac{\varphi \in \mathcal{L}^s(\Sigma;\tau[\mathsf{rec}\,\alpha.\tau/\alpha])}{[\mathsf{fold}]\varphi \in \mathcal{L}^s(\Sigma;\mathsf{rec}\,\alpha.\tau)} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma;\tau_1)}{[\mathsf{in}_1]\varphi \in \mathcal{L}^s(\Sigma;\tau_1 + \tau_2)} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma;\tau_2)}{[\mathsf{in}_2]\varphi \in \mathcal{L}^s(\Sigma;\tau_1 + \tau_2)}$$

(b) Modalities.

$$\frac{(p\colon \tau) \in \Sigma}{p \in \mathcal{L}^s(\Sigma;\tau)} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma;\tau) \quad p \notin \mathrm{dom}(\Sigma)}{\varphi \in \mathcal{L}^s(\Sigma, p\colon \sigma;\tau)} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma, p\colon \tau;\tau)}{(\mu^t p)\varphi \in \mathcal{L}^s(\Sigma;\tau)} \qquad \frac{\varphi \in \mathcal{L}^s(\Sigma, p\colon \tau;\tau)}{(\nu^t p)\varphi \in \mathcal{L}^s(\Sigma;\tau)}$$

(c) Bounded iteration, where t is an iteration term.

$$\frac{\varphi \in \mathcal{L}^+(\Sigma;\tau) \quad k\ \mathsf{Pos}\ \varphi}{(\exists k)\varphi \in \mathcal{L}^+(\Sigma;\tau)} \qquad \frac{\varphi \in \mathcal{L}^-(\Sigma;\tau) \quad k\ \mathsf{Neg}\ \varphi}{(\forall k)\varphi \in \mathcal{L}^-(\Sigma;\tau)} \qquad \frac{\psi \in \mathcal{L}^{-s}(;\sigma) \quad \varphi \in \mathcal{L}^s(;\tau)}{\psi \Vdash \varphi \in \mathcal{L}^s(;\sigma \to \tau)}$$

(d) Quantification and realizability implication, with $-s$ defined as $-\pm = \pm$, $-+ = -$ and $-- = +$.

Figure 2 Formation rules for $\mathcal{L}^s$ (the predicates Pos and Neg are defined in Figure 3).

3 The Logic

The main ingredient of this paper is the logic we use to annotate pure types when forming refinement types. This logic is in essence *manysorted*: for each pure type τ we have formulae φ *of type* τ, notation $\varphi \in \mathcal{L}^s(\tau)$, where s is $+$ for positive formulae and $-$ for a negative ones.

Positive and negative formulae are in essence closed under least and greatest fixpoints, respectively, and thus may contain fixpoint variables p. But least fixpoints $(\mu p)\varphi$ and greatest fixpoints $(\nu p)\psi$ are actually not directly available. They are defined as $(\exists k)(\mu^k p)\varphi$ and $(\forall \ell)(\nu^\ell p)\psi$, where $(\mu^k p)\varphi$ and $(\nu^\ell p)\psi$ represent bounded iterations $\varphi^k(\mathsf{False})$ and $\psi^\ell(\mathsf{True})$, respectively. This decomposition allows for a notion of *prenex* form (Proposition 3.9, § 3.2). A fixpoint logic with iteration variables was already considered in [36].

Formulae are presented in § 3.1, while deduction is discussed in § 3.2.

$$\frac{k \notin \mathrm{FV}(\varphi)}{k\ \mathsf{Pos}\ \varphi} \qquad \frac{k\ \mathsf{Pos}\ \varphi, \psi}{k\ \mathsf{Pos}\ \varphi \star \psi} \qquad \frac{k\ \mathsf{Pos}\ \varphi}{k\ \mathsf{Pos}\ [\triangle]\varphi} \qquad \frac{k\ \mathsf{Neg}\ \psi \qquad k\ \mathsf{Pos}\ \varphi}{k\ \mathsf{Pos}\ \psi \Vdash \varphi} \qquad \frac{k\ \mathsf{Pos}\ \varphi}{k\ \mathsf{Pos}\ (\exists \ell)\varphi}$$

$$\frac{k \notin \mathrm{FV}(\varphi)}{k\ \mathsf{Neg}\ \varphi} \qquad \frac{k\ \mathsf{Neg}\ \varphi, \psi}{k\ \mathsf{Neg}\ \varphi \star \psi} \qquad \frac{k\ \mathsf{Neg}\ \varphi}{k\ \mathsf{Neg}\ [\triangle]\varphi} \qquad \frac{k\ \mathsf{Pos}\ \psi \qquad k\ \mathsf{Neg}\ \varphi}{k\ \mathsf{Neg}\ \psi \Vdash \varphi} \qquad \frac{k\ \mathsf{Neg}\ \varphi}{k\ \mathsf{Neg}\ (\forall \ell)\varphi}$$

$$\frac{k\ \mathsf{Pos}\ \varphi}{k\ \mathsf{Pos}\ (\mu^t p)\varphi} \qquad \frac{k\ \mathsf{Pos}\ \varphi \qquad k \notin \mathrm{FV}(t)}{k\ \mathsf{Pos}\ (\nu^t p)\varphi} \qquad \frac{k\ \mathsf{Neg}\ \varphi}{k\ \mathsf{Neg}\ (\nu^t p)\varphi} \qquad \frac{k\ \mathsf{Neg}\ \varphi \qquad k \notin \mathrm{FV}(t)}{k\ \mathsf{Neg}\ (\mu^t p)\varphi}$$

Figure 3 The predicates Pos and Neg, where $\triangle \in \{\pi_1, \pi_2, \mathsf{in}_1, \mathsf{in}_2, \mathsf{fold}\}$, and where $\star$ is $\wedge$ or $\vee$.

3.1 Formulae

We assume given infinitely many *iteration variables* k, ℓ, etc. *Iteration terms*, intended to range over natural numbers, are given by the grammar

$$t \ ::= \ k \mid 0 \mid t + 1$$

We also assume given *fixpoint variables* p, q, etc. *Fixpoint contexts* Σ are lists of the form $p_1 : \sigma_1, \dots, p_n : \sigma_n$ where $p_i \neq p_j$ whenever $i \neq j$. We let $\mathrm{dom}(\Sigma)$ be $\{p_1, \dots, p_n\}$. Figure 2 gathers formation rules for sets $\mathcal{L}^s(\Sigma; \tau)$ indexed by fixpoint contexts and types. We write $\mathcal{L}^s(\tau)$ or $\mathcal{L}^s(; \tau)$ for $\mathcal{L}^s(\Sigma; \tau)$ when the fixpoint context Σ is empty.

▶ **Definition 3.1** (Formulae). *The sets $\mathcal{L}^{\pm}(\Sigma; \tau)$, $\mathcal{L}^{+}(\Sigma; \tau)$ and $\mathcal{L}^{-}(\Sigma; \tau)$ of respectively* neutral *$(\pm)$*, positive *$(+)$ and* negative *$(-)$* formulae *are defined by the rules in Figure 2 with $s \in \{\pm, +, -\}$, and using the predicates* Pos *and* Neg *defined in Figure 3.*

$\mathcal{L}^{\omega}(\tau)$ is the fragment of $\mathcal{L}^{\pm}(\tau)$ without *the rules in Figure 2c (but with $\Vdash$ from Figure 2d). $\mathcal{L}^{\wedge}(\tau)$ is the fragment of $\mathcal{L}^{\omega}(\tau)$ without binary disjunctions $\vee$ (but with* False*).*

Note that $\mathcal{L}^{\pm}$, $\mathcal{L}^{\omega}$ and $\mathcal{L}^{\wedge}$ are stable under the realizability implication $\Vdash$. In fact, $\mathcal{L}^{\pm}(\Sigma; \tau) = \mathcal{L}^{+}(\Sigma; \tau) \cap \mathcal{L}^{-}(\Sigma; \tau)$ and $\mathcal{L}^{\omega}(\tau) \subseteq \mathcal{L}^{\pm}(\tau)$ are defined by the rules in Figure 2a–2c and Figure 2a–2b respectively, together with

$$\frac{\psi \in \mathcal{L}^{\pm}(; \sigma) \qquad \varphi \in \mathcal{L}^{\pm}(; \tau)}{\psi \Vdash \varphi \in \mathcal{L}^{\pm}(; \sigma \to \tau)} \qquad \frac{\psi \in \mathcal{L}^{\omega}(\sigma) \qquad \varphi \in \mathcal{L}^{\omega}(\tau)}{\psi \Vdash \varphi \in \mathcal{L}^{\omega}(\sigma \to \tau)}$$

On the other hand, $\Vdash$ is contravariant in its first argument w.r.t. $+$ and $-$:

$$\frac{\psi \in \mathcal{L}^{-}(; \sigma) \qquad \varphi \in \mathcal{L}^{+}(; \tau)}{\psi \Vdash \varphi \in \mathcal{L}^{+}(; \sigma \to \tau)} \qquad \frac{\psi \in \mathcal{L}^{+}(; \sigma) \qquad \varphi \in \mathcal{L}^{-}(; \tau)}{\psi \Vdash \varphi \in \mathcal{L}^{-}(; \sigma \to \tau)} \tag{1}$$

In any case, $\varphi \Vdash \psi$ contains no free fixpoint variables. Also, formulae $\varphi \in \mathcal{L}^{\omega}(\tau)$ have no fixpoint variables nor iteration variables at all. When possible, we write $\mathcal{L}$ for $\mathcal{L}^s$.

Note that in Figure 3, we could have had a rule stating that k Pos $(\forall \ell)\varphi$ if k Pos φ. But this would have been useless for the left rule in Figure 2d, since a positive formula has no positive occurrence of a $(\forall \ell)$. A dual remark applies to k Neg $(\exists \ell)\varphi$.

The semantics of formulae is discussed in § 5.3. Their intended meaning is as follows. The formula $\psi \Vdash \varphi \in \mathcal{L}(\sigma \to \tau)$ is intended to select those $M : \sigma \to \tau$ such that φ holds on $MN : \tau$ whenever ψ holds on $N : \sigma$. The formula $[\mathsf{fold}]\varphi$ holds on M if, and only if, φ holds on $\mathsf{unfold}(M)$. For $i = 1, 2$, the formula $[\pi_i]\varphi$ selects those $M : \tau_1 \times \tau_2$ such that φ holds on $\pi_i(M)$, while $[\mathsf{in}_i]\varphi$ selects those $\mathsf{in}_j(M) : \tau_1 + \tau_2$ such that $j = i$ and φ holds on M. Hence $[\mathsf{in}_1]\mathsf{True} \wedge [\mathsf{in}_2]\mathsf{True}$ is a contradiction and $[\mathsf{in}_1]\mathsf{True} \vee [\mathsf{in}_2]\mathsf{True}$ fails on $\Omega_{\tau_1 + \tau_2}$.

▶ **Example 3.2.** We derive some composite modalities in $\mathcal{L}^\omega$ on the datatypes of Example 2.1. On Bool, we have formulae $[\mathsf{tt}] \coloneqq [\mathsf{in}_1][\langle\rangle]$ and $[\mathsf{ff}] \coloneqq [\mathsf{in}_2][\langle\rangle]$ which hold on $\mathsf{tt}, \mathsf{ff} \colon \mathsf{Bool}$, respectively. The formula $[\mathsf{tt}] \vee [\mathsf{ff}]$, which expresses totality at type Bool, fails on Ω_{Bool}.

On Nat, we have $[0] \coloneqq [\mathsf{fold}][\mathsf{in}_1][\langle\rangle]$ and a composite modality $[1+]\varphi \coloneqq [\mathsf{fold}][\mathsf{in}_2]\varphi$. By taking $[\mathsf{n}+1] \coloneqq [1+][\mathsf{n}]$, we have for each $n \in \mathbb{N}$ a formula $[\mathsf{n}] \in \mathcal{L}(\mathsf{Nat})$ which holds on $\mathsf{n} \colon \mathsf{Nat}$. We also have a composite modality $[\geq \mathsf{n}]\varphi$, defined as $[\geq 0]\varphi \coloneqq \varphi$ and $[\geq \mathsf{n}+1] \coloneqq [1+][\geq \mathsf{n}]\varphi$, such that $[\geq \mathsf{n}]\mathsf{True}$ holds on $\mathsf{inf} \colon \mathsf{Nat}$ and on $\mathsf{m} \colon \mathsf{Nat}$ for all $m \geq n$.

On streams $\mathsf{Stream}\,\sigma$, the composite modalities $[\mathsf{hd}]$ and $[\mathsf{tl}]$ are defined as $[\mathsf{hd}]\psi \coloneqq [\mathsf{fold}][\pi_1]\psi$ and $[\mathsf{tl}]\varphi \coloneqq [\mathsf{fold}][\pi_2]\varphi$. Given $\psi \in \mathcal{L}(\sigma)$ and $\varphi \in \mathcal{L}(\mathsf{Stream}\,\sigma)$, the formulae $[\mathsf{hd}]\psi \in \mathcal{L}(\mathsf{Stream}\,\sigma)$ and $[\mathsf{tl}]\varphi \in \mathcal{L}(\mathsf{Stream}\,\sigma)$ select those streams M such that ψ holds on $(\mathsf{hd}\,M)$ and such that φ holds on $(\mathsf{tl}\,M)$, respectively. We write $\bigcirc\varphi$ for $[\mathsf{tl}]\varphi$.

Similarly, on trees $\mathsf{Tree}\,\sigma$ one can define $[\mathsf{label}]$, $[\mathsf{left}]$ and $[\mathsf{right}]$ such that $[\mathsf{label}]\psi \in \mathcal{L}(\mathsf{Tree}\,\sigma)$ given $\psi \in \mathcal{L}(\sigma)$, and $[\mathsf{left}]\varphi, [\mathsf{right}]\varphi \in \mathcal{L}(\mathsf{Tree}\,\sigma)$ given $\varphi \in \mathcal{L}(\mathsf{Tree}\,\sigma)$. We also have composite modalities $\exists\bigcirc\varphi \coloneqq [\mathsf{left}]\varphi \vee [\mathsf{right}]\varphi$ and $\forall\bigcirc\varphi \coloneqq [\mathsf{left}]\varphi \wedge [\mathsf{right}]\varphi$ which allow for taking, respectively, disjunctions and conjunctions over the children of a node. ⌟

We now discuss some uses of fixpoint and iteration variables in $\mathcal{L}^\pm$, $\mathcal{L}^+$ and $\mathcal{L}^-$. Let $\varphi \in \mathcal{L}(\Sigma, p \colon \tau; \tau)$. Given $\psi \in \mathcal{L}(\Sigma; \tau)$ and $n \in \mathbb{N}$, we define $(p.\varphi)^n(\psi) \in \mathcal{L}(\Sigma; \tau)$ as

$$(p.\varphi)^0(\psi) \quad \coloneqq \quad \psi \qquad \text{and} \qquad (p.\varphi)^{n+1}(\psi) \quad \coloneqq \quad \varphi[(p.\varphi)^n(\psi)/p] \tag{2}$$

If t is a closed iteration term representing $n \in \mathbb{N}$ (i.e. t is the nth successor of 0), then the formulae $(\mu^t p)\varphi$ and $(\nu^t p)\varphi$ should be read as $(p.\varphi)^n(\mathsf{False})$ and $(p.\varphi)^n(\mathsf{True})$, respectively (cf Equation (5) in § 3.2 below). But in general, the iteration term t may contain free iteration variables. In fact, we shall see in Example 5.11 (§ 5.3) that the least fixpoint of $\varphi \in \mathcal{L}^+(p \colon \tau; \tau)$ and the greatest fixpoint of $\psi \in \mathcal{L}^-(p \colon \tau; \tau)$ can be represented respectively as

$$(\exists k)(\mu^k p)\varphi \in \mathcal{L}^+(\tau) \qquad \text{and} \qquad (\forall \ell)(\nu^\ell p)\psi \in \mathcal{L}^-(\tau) \tag{3}$$

▶ **Example 3.3.** The representation of fixpoints in Equation (3) allows for formulae expressing possibly unbounded behaviours. For instance, totality at type Nat is expressed by the *positive* formula $[\mathsf{tot}] \coloneqq (\exists k)(\mu^k p)([0] \vee [1+]p)$, intended to hold on all $\mathsf{n} \colon \mathsf{Nat}$ but not on $\mathsf{inf} \colon \mathsf{Nat}$.

On streams $\mathsf{Stream}\,\sigma$, the usual LTL-like *eventually* modality $\Diamond$ is available on *positive* formulae $\varphi \in \mathcal{L}^+(\mathsf{Stream}\,\sigma)$, namely $\Diamond\varphi \coloneqq (\exists k)(\mu^k p)(\varphi \vee \bigcirc p)$. The formula $\Diamond\varphi$ is intended to hold on those $M \colon \mathsf{Stream}\,\sigma$ such that φ holds on $(\mathsf{tl}^n\,M)$ for some $n \in \mathbb{N}$.

Regarding greatest fixpoints, the CTL-like modalities $\forall\Box$ and $\exists\Box$ on $\mathsf{Tree}\,\sigma$ are available for *negative* formulae: given $\psi \in \mathcal{L}^-(\mathsf{Tree}\,\sigma)$, we let $\forall\Box\psi \coloneqq (\forall\ell)(\nu^\ell p)(\psi \wedge \forall\bigcirc p)$ and $\exists\Box\psi \coloneqq (\forall\ell)(\nu^\ell p)(\psi \wedge \exists\bigcirc p)$. The intended meaning of $\forall\Box[\mathsf{label}]\psi \in \mathcal{L}^-(\mathsf{Tree}\,\sigma)$ is to select those trees of σ's whose node labels all satisfy $\psi \in \mathcal{L}^-(\sigma)$, while $\exists\Box[\mathsf{label}]\psi \in \mathcal{L}^-(\mathsf{Tree}\,\sigma)$ asks ψ to hold on all labels in some infinite path.

We thus obtain for each $n \in \mathbb{N}$ the following positive formula

$$\forall\Box[\mathsf{label}]([\mathsf{tt}] \vee [\mathsf{ff}]) \;\wedge\; \exists\bigcirc\exists\Box[\mathsf{label}][\mathsf{tt}] \;\wedge\; \exists\bigcirc\exists\Box[\mathsf{label}][\mathsf{ff}] \quad \Vdash\!\!\mapsto \quad \Diamond[\mathsf{hd}][\geq \mathsf{n}][\mathsf{tot}] \tag{4}$$

in $\mathcal{L}^+(\mathsf{Tree}\,\mathsf{Bool} \to \mathsf{Stream}\,\mathsf{Nat})$, which expresses the specification for $\mathsf{count} \circ \mathsf{bft}$ devised in § 1: Given a totally defined tree of Booleans, if the children of the root have an infinite path with only tt's and an infinite path with only ff's, then $\mathsf{count} \circ \mathsf{bft}$ produces a stream which contains a totally defined number $\geq n$. Note that fixpoints of the same polarity can be nested, e.g. $\Diamond[\mathsf{hd}][\geq \mathsf{n}][\mathsf{tot}] \in \mathcal{L}^+(\mathsf{Stream}\,\mathsf{Nat})$ with $[\mathsf{tot}] \in \mathcal{L}^+(\mathsf{Nat})$. ⌟

▶ Remark 3.4 (Non-Examples). Given a *negative* $\psi \in \mathcal{L}^-(\mathsf{Stream}\,\sigma)$, we have the usual *always* modality $\Box\psi \coloneqq (\forall\ell)(\nu^\ell p)(\psi \wedge \bigcirc p) \in \mathcal{L}^-(\mathsf{Stream}\,\sigma)$ which holds on M if ψ holds on $(\mathsf{tl}^n\, M)$ for all $n \in \mathbb{N}$. A composite $\Box\Diamond[\mathsf{hd}]\varphi$ would require a stream to have infinitely many elements satisfying φ. But $\Box\Diamond[\mathsf{hd}]\varphi$ is not a formula since $\Diamond[\mathsf{hd}]\varphi$ is not negative.

Also, in order to universally quantify over $n \in \mathbb{N}$ in the r.-h.s. of (4), one could think of $(\forall\ell)\Diamond[\mathsf{hd}](\mu^\ell p)([\mathsf{tot}] \vee [1{+}]p)$. This is not a formula since ψ has to be negative in $(\forall\ell)\psi$. ⌟

3.2 Deduction

Deduction will enter the refinement type system via subtyping (Figure 6 in § 4).

▶ **Definition 3.5** (Deduction). *A* sequent *has the form* $\psi \vdash \varphi$ *where* $\varphi, \psi \in \mathcal{L}^s(\Sigma;\tau)$. *The* derivable *sequents are defined by the rules in Figure 4, using the* consistency predicate $\mathcal{C}$ *defined in Figure 5. Write* $\psi \dashv\vdash \varphi$ *when the sequents* $\psi \vdash \varphi$ *and* $\varphi \vdash \psi$ *are both derivable.*

We discuss some rules in Figure 4, heading to basic properties of the deduction system. We begin with a notion of normal form for $\mathcal{L}^\omega$. Let $\mathcal{L}^{\vee\wedge}(\tau)$ be the closure of $\mathcal{L}^\wedge(\tau)$ under disjunctions (but under no other rules in Figure 2, so e.g. $[\pi_1](\varphi \vee \psi)$ is *not* in $\mathcal{L}^{\vee\wedge}$).

As usual, the converse of the distributive rule (D) in Figure 4a is derivable, and so is the dual law $\psi \vee (\varphi \wedge \theta) \dashv\vdash (\psi \vee \varphi) \wedge (\psi \vee \theta)$. Moreover, given $\triangle \in \{\pi_i, \mathsf{in}_i, \mathsf{fold}\}$ we can derive $[\triangle](\varphi \vee \psi) \dashv\vdash [\triangle]\varphi \vee [\triangle]\psi$ (and similarly for $\wedge$). It follows that for each $\varphi \in \mathcal{L}^{\vee\wedge}$, there is some $\psi \in \mathcal{L}^{\vee\wedge}$ such that $[\triangle]\varphi \dashv\vdash \psi$. Using the rules $(\vee/\|\!\!\to)$ and $(\|\!\!\to/\vee)$ in Figure 4e, we can similarly obtain that for each $\varphi_1, \varphi_2 \in \mathcal{L}^{\vee\wedge}$ there is some $\psi \in \mathcal{L}^{\vee\wedge}$ such that $(\varphi_1 \mathrel{\|\!\!\to} \varphi_2) \dashv\vdash \psi$ (but beware that the rule $(\|\!\!\to/\vee)$ requires δ to be in $\mathcal{L}^\wedge$). Hence:

▶ **Lemma 3.6.** *For each* $\varphi \in \mathcal{L}^\omega(\tau)$, *there is some* $\psi \in \mathcal{L}^{\vee\wedge}(\tau)$ *such that* $\varphi \dashv\vdash \psi$.

▶ Remark 3.7. The rule (C) in Figure 4e is a kind of a zero-ary case of $(\|\!\!\to/\vee)$. Since $\mathsf{True} \vdash (\mathsf{False} \mathrel{\|\!\!\to} \mathsf{False})$, the sequent $(\psi \mathrel{\|\!\!\to} \mathsf{False}) \vdash \mathsf{False}$ is derivable only when ψ is consistent. This is enforced by the consistency predicate $\mathcal{C}(\psi)$, see Theorem 5.13(1) in § 5.3. Our predicate $\mathcal{C}$ is akin to [1, 32], but differs from the *coprimeness* predicates of [2, 7, 4]. Note that the clauses defining $\mathcal{C}$ are positive (compare with [7, Figure 3] and [4, Figure 10.3]).

At sum types, we have inconsistent formulae which do not mention False: The rule (I) in Figure 4b means that it is inconsistent to assert both $[\mathsf{in}_1]\mathsf{True}$ and $[\mathsf{in}_2]\mathsf{True}$. Hence, our sum types are *disjoint*, and we do not need the termination predicates of [1, 2].

Besides, note that the sequent $\mathsf{True} \vdash [\mathsf{in}_1]\mathsf{True} \vee [\mathsf{in}_2]\mathsf{True}$ is *not* derivable. ⌟

If t is a closed iteration term, then using the rules in Figure 4d we can derive

$$(\mu^t p)\varphi \quad\dashv\vdash\quad (p.\varphi)^n(\mathsf{False}) \qquad \text{and} \qquad (\nu^t p)\varphi \quad\dashv\vdash\quad (p.\varphi)^n(\mathsf{True}) \tag{5}$$

where t is the nth successor of 0 (cf Equation (2) in § 3.1). This yields:

▶ **Lemma 3.8.** *For each* closed $\varphi \in \mathcal{L}^\pm(\tau)$, *there is some* $\psi \in \mathcal{L}^\omega(\tau)$ *such that* $\varphi \dashv\vdash \psi$.

The main fact of this § 3.2 is that quantifiers can be assumed to be in prenex position. Quantifiers of the same type can be merged using the rules (∃M) and (∀M) in Figure 4c, themselves permitted by the polarity constraints in Figure 2d. Formally, the *positive*, resp. *negative*, *prenex forms* are the formulae $(\exists k)\psi$, resp. $(\forall k)\psi$, with $\psi \in \mathcal{L}^\pm$ neutral.

The rules (WF), (CC) in Figure 4e, which apply to positive formulae, are crucial to obtain prenex forms. Note also that the *Frobenius* rules (∃L) and (∀R) in Figure 4c make it possible to derive the following when $k \notin \mathrm{FV}(\psi)$:

$$\psi \wedge (\exists k)\varphi \quad\dashv\vdash\quad (\exists k)(\psi \wedge \varphi) \qquad \text{and} \qquad (\forall k)(\varphi \vee \psi) \quad\dashv\vdash\quad (\forall k)\varphi \vee \psi$$

▶ **Proposition 3.9.** *For each* $\varphi \in \mathcal{L}^s(\Sigma;\tau)$, *there is a prenex* $\psi \in \mathcal{L}^s(\Sigma;\tau)$ *such that* $\varphi \dashv\vdash \psi$.

$$\frac{}{\varphi \vdash \varphi} \qquad \frac{\psi \vdash \theta \qquad \theta \vdash \varphi}{\psi \vdash \varphi} \qquad (\mathsf{D})\frac{}{\psi \wedge (\varphi \vee \theta) \vdash (\psi \wedge \varphi) \vee (\psi \wedge \theta)}$$

$$\frac{\psi \vdash \varphi_1 \qquad \psi \vdash \varphi_2}{\psi \vdash \varphi_1 \wedge \varphi_2} \qquad \frac{\psi_1 \vdash \varphi}{\psi_1 \wedge \psi_2 \vdash \varphi} \qquad \frac{\psi_1 \vdash \varphi}{\psi_1 \wedge \psi_2 \vdash \varphi} \qquad \frac{}{\psi \vdash \mathsf{True}}$$

$$\frac{\psi \vdash \varphi_1}{\psi \vdash \varphi_1 \vee \varphi_2} \qquad \frac{\psi \vdash \varphi_2}{\psi \vdash \varphi_1 \vee \varphi_2} \qquad \frac{\psi_1 \vdash \varphi \qquad \psi_2 \vdash \varphi}{\psi_1 \vee \psi_2 \vdash \varphi} \qquad \frac{}{\mathsf{False} \vdash \varphi}$$

(a) Propositional rules.

$$(\mathsf{I})\frac{}{[\mathsf{in}_1]\varphi \wedge [\mathsf{in}_2]\psi \vdash \mathsf{False}} \qquad \frac{\psi \vdash \varphi}{[\triangle]\psi \vdash [\triangle]\varphi} \qquad \frac{\psi' \vdash \psi \qquad \varphi \vdash \varphi'}{\psi \Vdash \varphi \vdash \psi' \Vdash \varphi'}$$

$$\frac{}{[\triangle]\varphi_1 \wedge [\triangle]\varphi_2 \vdash [\triangle](\varphi_1 \wedge \varphi_2)} \qquad \frac{\triangle \notin \{\mathsf{in}_1, \mathsf{in}_2\}}{\mathsf{True} \vdash [\triangle]\mathsf{True}} \qquad \frac{}{(\forall k)[\triangle]\varphi \vdash [\triangle](\forall k)\varphi}$$

$$\frac{}{[\triangle](\varphi_1 \vee \varphi_2) \vdash [\triangle]\varphi_1 \vee [\triangle]\varphi_2} \qquad \frac{}{[\triangle]\mathsf{False} \vdash \mathsf{False}} \qquad \frac{}{[\triangle](\exists k)\varphi \vdash (\exists k)[\triangle]\varphi}$$

(b) Modal rules, where $\triangle \in \{\pi_1, \pi_2, \mathsf{in}_1, \mathsf{in}_2, \mathsf{fold}\}$.

$$\frac{\psi \vdash \varphi[t/k]}{\psi \vdash (\exists k)\varphi} \qquad (\exists\mathsf{L})\frac{\theta \wedge \varphi \vdash \psi \qquad k \notin \mathrm{FV}(\psi, \theta)}{\theta \wedge (\exists k)\varphi \vdash \psi} \qquad (\exists\mathsf{M})\frac{}{(\exists k)(\exists \ell)\varphi \vdash (\exists k)\varphi[k/\ell]}$$

$$\frac{\varphi[t/k] \vdash \psi}{(\forall k)\varphi \vdash \psi} \qquad (\forall\mathsf{R})\frac{\psi \vdash \varphi \vee \theta \qquad k \notin \mathrm{FV}(\psi, \theta)}{\psi \vdash (\forall k)\varphi \vee \theta} \qquad (\forall\mathsf{M})\frac{}{(\forall k)\varphi[k/\ell] \vdash (\forall k)(\forall \ell)\varphi}$$

$$(\iota/\exists)\frac{k \notin \mathrm{FV}(t)}{(\iota^t p)(\exists k)\varphi \vdash (\exists k)(\iota^t p)\varphi} \qquad (\forall/\iota)\frac{k \notin \mathrm{FV}(t)}{(\forall k)(\iota^t p)\varphi \vdash (\iota^t p)(\forall k)\varphi}$$

(c) Quantifier rules, where $\iota \in \{\mu, \nu\}$.

$$\frac{}{(\mu^0 p)\varphi \vdash \mathsf{False}} \qquad \frac{(\mu^t p)\varphi \vdash \psi}{(\mu^{t+1} p)\varphi \vdash \varphi[\psi/p]} \qquad \frac{}{\varphi[(\mu^t p)\varphi/p] \vdash (\mu^{t+1} p)\varphi} \qquad \frac{\psi \vdash \varphi}{(\mu^t p)\psi \vdash (\mu^t p)\varphi}$$

$$\frac{}{\mathsf{True} \vdash (\nu^0 p)\varphi} \qquad \frac{\psi \vdash (\nu^t p)\varphi}{\varphi[\psi/p] \vdash (\nu^{t+1} p)\varphi} \qquad \frac{}{(\nu^{t+1} p)\varphi \vdash \varphi[(\nu^t p)\varphi/p]} \qquad \frac{\psi \vdash \varphi}{(\nu^t p)\psi \vdash (\nu^t p)\varphi}$$

(d) Bounded iteration.

$$(\mathsf{WF})\frac{k \notin \mathrm{FV}(\varphi)}{(\forall k)\psi \Vdash \varphi \vdash (\exists k)(\psi \Vdash \varphi)} \qquad (\mathsf{CC})\frac{\psi \in \mathcal{L}^{\pm} \qquad k \notin \mathrm{FV}(\psi)}{\psi \Vdash (\exists k)\varphi \vdash (\exists k)(\psi \Vdash \varphi)}$$

$$(\exists/\Vdash)\frac{k \notin \mathrm{FV}(\varphi)}{(\forall k)(\psi \Vdash \varphi) \vdash (\exists k)\psi \Vdash \varphi} \qquad (\Vdash/\forall)\frac{k \notin \mathrm{FV}(\psi)}{(\forall k)(\psi \Vdash \varphi) \vdash \psi \Vdash (\forall k)\varphi}$$

$$(\Vdash/\wedge)\frac{}{(\psi \Vdash \varphi_1) \wedge (\psi \Vdash \varphi_2) \vdash \psi \Vdash (\varphi_1 \wedge \varphi_2)} \qquad \frac{}{\mathsf{True} \vdash (\psi \Vdash \mathsf{True})}$$

$$(\vee/\Vdash)\frac{}{(\psi_1 \Vdash \varphi) \wedge (\psi_1 \Vdash \varphi) \vdash (\psi_1 \vee \psi_2) \Vdash \varphi} \qquad \frac{}{\mathsf{True} \vdash (\mathsf{False} \Vdash \varphi)}$$

$$(\Vdash/\vee)\frac{\delta \in \mathcal{L}^{\wedge}}{\delta \Vdash (\varphi_1 \vee \varphi_2) \vdash (\delta \Vdash \varphi_1) \vee (\delta \Vdash \varphi_2)} \qquad (\mathsf{C})\frac{\mathcal{C}(\delta) \qquad \delta \in \mathcal{L}^{\wedge}}{(\delta \Vdash \mathsf{False}) \vdash \mathsf{False}}$$

(e) Rules for the realizability implication $\Vdash$ (the predicate $\mathcal{C}$ is defined in Figure 5).

Figure 4 Deduction rules.

$$\frac{}{\mathcal{C}([\langle\rangle])} \qquad \frac{\mathcal{C}(\varphi)}{\mathcal{C}([\mathsf{fold}]\varphi)} \qquad \frac{\mathcal{C}(\varphi) \quad \mathcal{C}(\psi)}{\mathcal{C}([\pi_1]\varphi \wedge [\pi_2]\psi)} \qquad \frac{\mathcal{C}(\varphi)}{\mathcal{C}([\mathsf{in}_1]\varphi)} \qquad \frac{\mathcal{C}(\varphi)}{\mathcal{C}([\mathsf{in}_2]\varphi)}$$

$$\frac{}{\mathcal{C}(\mathsf{True})} \qquad \frac{\mathcal{C}(\psi) \quad \psi \vdash \varphi \quad \varphi \in \mathcal{L}^{\wedge}}{\mathcal{C}(\varphi)} \qquad \frac{\begin{array}{c} I \text{ finite and } \forall i \in I,\ \mathcal{C}(\psi_i) \text{ and } \mathcal{C}(\varphi_i); \\ \forall J \subseteq I,\ \bigwedge_{j\in J} \psi_j \vdash \mathsf{False} \ \text{ or } \ \mathcal{C}\left(\bigwedge_{j\in J} \varphi_j\right) \end{array}}{\mathcal{C}\left(\bigwedge_{i\in I}(\psi_i \Vdash \varphi_i)\right)}$$

Figure 5 The consistency predicate $\mathcal{C}$.

$$\frac{}{T \preceq T} \qquad \frac{T \preceq U \quad U \preceq V}{T \preceq V} \qquad \frac{}{T \preceq |T|} \qquad \frac{}{\tau \preceq \{\tau \mid \mathsf{True}\}} \qquad \frac{\psi \vdash \varphi}{\{\tau \mid \psi\} \preceq \{\tau \mid \varphi\}}$$

$$\frac{T \preceq T' \quad U \preceq U'}{T \times U \preceq T' \times U'} \qquad \frac{}{\{\tau \mid \varphi\} \times \{\sigma \mid \psi\} \simeq \{\tau \times \sigma \mid [\pi_1]\varphi \wedge [\pi_2]\psi\}}$$

$$\frac{U' \preceq U \quad T \preceq T'}{U \to T \preceq U' \to T'} \qquad \frac{}{\{\sigma \mid \psi\} \to \{\tau \mid \varphi\} \simeq \{\sigma \to \tau \mid \psi \Vdash \varphi\}}$$

Figure 6 Subtyping.

4 The Refinement Type System

We consider *neutral*, *positive* and *negative refinement types* (or *types*), notation T^s, U^s, etc. with $s = \pm, +, -$ respectively. They are given by the grammar

$$T^s, U^s \ ::= \ \tau \ \mid \ \{\tau \mid \varphi\} \ \mid \ T^s \times U^s \ \mid \ U^{-s} \to T^s$$

where τ is a pure type and where $\varphi \in \mathcal{L}^s(\tau)$ is *closed*. When possible, we write T for T^s.

▶ **Example 4.1.** For each $n \in \mathbb{N}$, the specification for $\mathsf{count} \circ \mathsf{bft}$ is given either by the positive refinement type $\{\mathsf{Tree\,Bool} \to \mathsf{Stream\,Nat} \mid \Phi^+\}$, where Φ^+ is the formula in Equation (4), Example 3.3, or by the positive refinement type S^+ defined as

$$\left\{ \mathsf{Tree\,Bool} \;\middle|\; \forall\Box[\mathsf{label}]([\mathsf{tt}] \vee [\mathsf{ff}]) \ \wedge \ \exists\bigcirc\exists\Box[\mathsf{label}][\mathsf{tt}] \ \wedge \ \exists\bigcirc\exists\Box[\mathsf{label}][\mathsf{ff}] \right\} \longrightarrow \left\{ \mathsf{Stream\,Nat} \;\middle|\; \Diamond[\mathsf{hd}][\geq \mathsf{n}][\mathsf{tot}] \right\}$$

⌟

We shall consider typing judgements of the form $\mathcal{E} \vdash M : T$, where $\mathcal{E}$ is allowed to mention refinement types. A judgement $M : \{\tau \mid \varphi\}$ is intended to mean that M is of pure type τ and satisfies φ. *Positive* typing judgements are of the form $\mathcal{E}^- \vdash M : T^+$, where only negative types are declared in $\mathcal{E}^-$.

We derive typing judgements $\mathcal{E} \vdash M : T$ using the rules in Figure 7 augmented with those in Figure 1 (§ 2). Deduction on formulae (§ 3.2) enters the type system in Figure 7c via the subtyping relation $U \preceq T$ defined in Figure 6, where $U \simeq T$ stands for the conjunction of $U \preceq T$ and $T \preceq U$. Note that for each τ we have $\tau \simeq \{\tau \mid \mathsf{True}\}$. Subtyping is extended to typing contexts: given $\mathcal{E} = x_1 : U_1, \ldots, x_n : U_n$ and $\mathcal{E}' = x_1 : U'_1, \ldots, x_n : U'_n$, we let $\mathcal{E} \preceq \mathcal{E}'$ when $U_i \preceq U'_i$ for all $i = 1, \ldots, n$. Note that if $\mathcal{E} \vdash M : T$ is derivable then so is $|\mathcal{E}| \vdash M : |T|$, where the *underlying pure type* $|T|$ of T is defined by induction from $|\tau| := \tau$ and $|\{\tau \mid \varphi\}| := \tau$, and where $|\mathcal{E}|$ is the extension of $|-|$ to $\mathcal{E}$.

The rules in Figures 6 and 7 are adaptations of those in [2, 7, 19, 32]. In particular, the crucial rule for $\mathsf{fix}\, x.M$ in Figure 7d comes from [2].

$$\frac{(x\colon T)\in\mathcal{E}}{\mathcal{E}\vdash x\colon T}\qquad\frac{\mathcal{E},\,x\colon U\vdash M\colon T}{\mathcal{E}\vdash\lambda x.M\colon U\to T}\qquad\frac{\mathcal{E}\vdash M\colon U\to T\qquad\mathcal{E}\vdash N\colon U}{\mathcal{E}\vdash MN\colon T}$$

$$\frac{\mathcal{E}\vdash M\colon T\qquad\mathcal{E}\vdash N\colon U}{\mathcal{E}\vdash\langle M,N\rangle\colon T\times U}\qquad\frac{\mathcal{E}\vdash M\colon T\times U}{\mathcal{E}\vdash\pi_1(M)\colon T}\qquad\frac{\mathcal{E}\vdash M\colon T\times U}{\mathcal{E}\vdash\pi_2(M)\colon U}$$

(a) Basic rules.

$$\frac{\mathcal{E}\vdash M\colon\{\tau\mid\varphi_1\}\qquad\mathcal{E}\vdash M\colon\{\tau\mid\varphi_2\}}{\mathcal{E}\vdash M\colon\{\tau\mid\varphi_1\wedge\varphi_2\}}\qquad\frac{|\mathcal{E}|,\,x\colon\sigma,\,|\mathcal{E}'|\vdash M\colon|T|}{\mathcal{E},\,x\colon\{\sigma\mid\mathsf{False}\},\,\mathcal{E}'\vdash M\colon T}$$

$$\frac{\mathcal{E},\,x\colon\{\sigma\mid\psi_1\},\,\mathcal{E}'\vdash M\colon T\qquad\mathcal{E},\,x\colon\{\sigma\mid\psi_2\},\,\mathcal{E}'\vdash M\colon T}{\mathcal{E},\,x\colon\{\sigma\mid\psi_1\vee\psi_2\},\,\mathcal{E}'\vdash M\colon T}$$

(b) Propositional connectives.

$$\frac{\mathcal{E}'\preceq\mathcal{E}\qquad T\preceq T'\qquad\mathcal{E}\vdash M\colon T}{\mathcal{E}'\vdash M\colon T'}$$

(c) Subtyping (cf Figure 6).

$$\frac{\mathcal{E}\vdash\mathsf{fix}\,x.M\colon\{\tau\mid\psi\}\qquad\mathcal{E},x\colon\{\tau\mid\psi\}\vdash M\colon\{\tau\mid\varphi\}}{\mathcal{E}\vdash\mathsf{fix}\,x.M\colon\{\tau\mid\varphi\}}$$

(d) Fixpoints.

$$\frac{\mathcal{E}\vdash M\colon\{\tau_1\times\tau_2\mid[\pi_i]\varphi\}}{\mathcal{E}\vdash\pi_i(M)\colon\{\tau_i\mid\varphi\}}\qquad\frac{\mathcal{E}\vdash M_i\colon\{\tau_i\mid\varphi\}\qquad\mathcal{E}\vdash M_{3-i}\colon\tau_{3-i}}{\mathcal{E}\vdash\langle M_1,M_2\rangle\colon\{\tau_1\times\tau_2\mid[\pi_i]\varphi\}}$$

$$\frac{}{\mathcal{E}\vdash\langle\rangle\colon\{\mathsf{Unit}\mid[\langle\rangle]\}}\qquad\frac{\mathcal{E}\vdash M\colon\{\tau_i\mid\varphi\}}{\mathcal{E}\vdash\mathsf{in}_i(M)\colon\{\tau_1+\tau_2\mid[\mathsf{in}_i]\varphi\}}$$

$$\frac{\mathcal{E}\vdash M\colon\{\tau_1+\tau_2\mid[\mathsf{in}_i]\varphi\}\qquad\mathcal{E},x_i\colon\{\tau_i\mid\varphi\}\vdash N_i\colon T\qquad|\mathcal{E}|,x_{3-i}\colon\tau_{3-i}\vdash N_{3-i}\colon|T|}{\mathcal{E}\vdash\mathsf{case}\;M\;[x_1\mapsto N_1\mid x_2\mapsto N_2]\colon T}$$

$$\frac{\mathcal{E}\vdash M\colon\{\tau[\mathsf{rec}\,\alpha.\tau/\alpha]\mid\varphi\}}{\mathcal{E}\vdash\mathsf{fold}(M)\colon\{\mathsf{rec}\,\alpha.\tau\mid[\mathsf{fold}]\varphi\}}\qquad\frac{\mathcal{E}\vdash M\colon\{\mathsf{rec}\,\alpha.\tau\mid[\mathsf{fold}]\varphi\}}{\mathcal{E}\vdash\mathsf{unfold}(M)\colon\{\tau[\mathsf{rec}\,\alpha.\tau/\alpha]\mid\varphi\}}$$

(e) Modalities, where $i\in\{1,2\}$.

Figure 7 Typing rules.

▶ **Example 4.2.** The bottom $\simeq$-rule in Figure 6 yields $S^+\simeq\{\mathsf{Tree\,Bool}\to\mathsf{Stream\,Nat}\mid\Phi^+\}$ in Example 4.1, as well as the following derived typing rules.

$$\frac{\mathcal{E},x\colon\{\sigma\mid\psi\}\vdash M\colon\{\tau\mid\varphi\}}{\mathcal{E}\vdash\lambda x.M\colon\{\sigma\to\tau\mid\psi\Vdash\varphi\}}\qquad\frac{\mathcal{E}\vdash M\colon\{\sigma\to\tau\mid\psi\Vdash\varphi\}\qquad\mathcal{E}\vdash N\colon\{\sigma\mid\psi\}}{\mathcal{E}\vdash MN\colon\{\tau\mid\varphi\}}$$

▶ **Lemma 4.3.** *For each type T^s, there is a $\varphi\in\mathcal{L}^s(|T|)$ such that $T^s\simeq\{|T^s|\mid\varphi\}$.*

▶ Remark 4.4. Refinement types are *not* closed under $+$ since in view of Remark 3.7, the equivalence $\{\tau\mid\varphi\}+\{\sigma\mid\psi\}\simeq\{\tau+\sigma\mid[\mathsf{in}_1]\varphi\vee[\mathsf{in}_2]\psi\}$ would be wrong, thus preventing from Lemma 4.3 to hold. ⌟

Our main result (Theorem 5.17) is that the *positive* fragment of this type system is (sound and) complete w.r.t. the usual Scott semantics: given $\vdash M\colon\tau$ and a *positive* $\varphi\in\mathcal{L}^+(\tau)$,

$$\vdash M\colon\{\tau\mid\varphi\}\qquad\text{if, and only if,}\qquad\varphi\text{ holds on }[\![M]\!]\text{ in the Scott semantics.}$$

In particular, the judgement $\vdash\mathsf{count}\circ\mathsf{bft}\colon S^+$ is derivable.

5 Semantics

We interpret pure types as Scott domains and λ-terms as Scott-continuous functions (§ 5.1). Using the well-known fact that Scott domains are spectral spaces (§ 5.2), this yields the semantics of our logic (§ 5.3) and of our refinement type system (§ 5.4). Our main result is the Positive Completeness Theorem 5.17 in § 5.4.

We mostly use the terminology in [4, §1]. A *dcpo* is a poset with suprema of all directed subsets (a subset D of a poset is *directed* if all (possibly empty) finite $F \subseteq D$ have an upper bound in D; directed sets are non-empty). A *cpo* is a dcpo with a least element (often denoted $\bot$). A function between dcpos is *Scott-continuous* if it preserves the order (i.e. is monotone) as well as directed suprema.

▶ **Definition 5.1** (Scott Domain). *A* Scott domain *is a bounded-complete algebraic cpo.* **Scott** *is the category of Scott domains and Scott-continuous functions.*

Recall that a cpo X is bounded-complete if any two $x, y \in X$ have a sup (or *least* upper bound) $x \vee y \in X$ whenever they have an upper bound.

An element x of a dcpo X is *finite* if for all directed $D \subseteq X$ such that $x \leq \bigvee D$, we have $x \leq d$ for some $d \in D$ (finite elements are called *compact* in [4]). Note that $\bot$ is always finite, and that if $d, d' \in X$ are finite, then $d \vee d'$ is finite whenever it exists. A dcpo X is *algebraic* if for each $x \in X$, the set $\{d \in X \mid d \text{ finite and } \leq x\}$ is directed and has sup x.

The category **Scott** is Cartesian-closed (see e.g. [3, Corollary 4.1.6]).

5.1 Semantics of the Pure System

Typed terms $\mathcal{E} \vdash M : \tau$ of the pure system (§ 2) are interpreted as morphisms $[\![M]\!] \colon [\![\mathcal{E}]\!] \to [\![\tau]\!]$ in **Scott**, where $[\![\mathcal{E}]\!] = \prod_{i=1}^{n} [\![\sigma_i]\!]$ when $\mathcal{E} = x_1 : \sigma_1, \dots, x_n : \sigma_n$. See [31, §C] for details.

Product types $\tau \times \sigma$ are interpreted using the Cartesian product of **Scott**, i.e. the Cartesian product of sets equipped with component-wise order. Arrow types $\sigma \to \tau$ are interpreted using the closed structure of **Scott**, given by equipping each homset $\mathbf{Scott}(X, Y)$ with the pointwise order.

The interpretation of Unit is $\{\bot, \top\}$ with $[\![\langle\rangle]\!] \coloneqq \top$. We let $[\![\tau + \sigma]\!]$ be the *weak* coproduct $([\![\tau]\!] \amalg [\![\sigma]\!])_\bot$, i.e., the disjoint union of $[\![\tau]\!]$ and $[\![\sigma]\!]$ augmented with a new least element $\bot$ (**Scott** does not have finite coproducts [18]). We refer to [4, 3, 37] for the interpretation of recursive types $\mathsf{rec}\,\alpha.\tau$. Term-level fixpoints $\mathsf{fix}\,x.M$ are interpreted using the usual fixpoint combinators $\mathsf{Y} \colon (X \to X) \to X$ taking $f \colon X \to X$ to $\mathsf{Y}(f) \coloneqq \bigvee_{n \in \mathbb{N}} f^n(\bot)$.

▶ **Example 5.2.** The domain $[\![\mathsf{Stream}\,\sigma]\!]$ of streams is $[\![\sigma]\!]^{\mathbb{N}}$ equipped with the pointwise order, while the domain $[\![\mathsf{Tree}\,\sigma]\!]$ of trees is $[\![\sigma]\!]^{\{0,1\}^*}$. The finite elements are those $z \in [\![\sigma]\!]^I$ such that $z(p)$ is finite in $[\![\sigma]\!]$ for all $p \in I$, and $z(p) \neq \bot$ for at most finitely many $p \in I$.

Given $x \in [\![\mathsf{Stream}\,\sigma]\!]$ we have $[\![\mathsf{hd}]\!](x) = x(0)$, while $[\![\mathsf{tl}]\!](x)$ is the stream taking $n \in \mathbb{N}$ to $x(n+1) \in [\![\sigma]\!]$. Moreover, $x = [\![\mathsf{Cons}]\!]([\![\mathsf{hd}]\!](x), [\![\mathsf{tl}]\!](x))$. Similarly, if $y \in [\![\mathsf{Tree}\,\sigma]\!]$ then $[\![\mathsf{label}]\!](y) = y(\varepsilon)$ is the root label of y, while $[\![\mathsf{left}]\!](y)$ and $[\![\mathsf{right}]\!](y)$ are the left- and right-subtrees of y, respectively.

One can then check that $[\![\mathsf{bft}]\!] \colon [\![\mathsf{Tree}\,\sigma]\!] \to [\![\mathsf{Stream}\,\sigma]\!]$ indeed computes a breadth-first tree traversal (see [31, §C.3.1]). In particular, $[\![\mathsf{count}]\!] \circ [\![\mathsf{bft}]\!]$ satisfies its expected specification (cf Equation (4) in Example 3.3). ⌟

5.2 Scott Domains as Spectral Spaces

The semantics of formulae and refinement types involves some topology.

$$\begin{array}{lll}
[\![\varphi \wedge \psi]\!]\rho \coloneqq [\![\varphi]\!]\rho \cap [\![\psi]\!]\rho & [\![p]\!]\rho \coloneqq \rho(p) & [\![(\mu^t p)\varphi]\!]\rho \coloneqq [\![(p.\varphi)^{[\![t]\!]\rho}(\mathsf{False})]\!]\rho \\
[\![\varphi \vee \psi]\!]\rho \coloneqq [\![\varphi]\!]\rho \cup [\![\psi]\!]\rho & [\![[\langle\rangle]]\!]\rho \coloneqq \{\top\} & [\![(\nu^t p)\varphi]\!]\rho \coloneqq [\![(p.\varphi)^{[\![t]\!]\rho}(\mathsf{True})]\!]\rho \\
[\![\mathsf{True}]\!]\rho \coloneqq [\![\tau]\!] & [\![[\triangle]\varphi]\!]\rho \coloneqq [\![[\triangle]]\!]([\![\varphi]\!]\rho) & [\![(\exists k)\varphi]\!]\rho \coloneqq \bigcup_{n\in\mathbb{N}} [\![\varphi]\!]\rho[n/k] \\
[\![\mathsf{False}]\!]\rho \coloneqq \emptyset & [\![\psi \mathrel{|\!\!\mapsto} \varphi]\!]\rho \coloneqq [\![\psi]\!]\rho \mathrel{|\!\!\mapsto} [\![\varphi]\!]\rho & [\![(\forall k)\varphi]\!]\rho \coloneqq \bigcap_{n\in\mathbb{N}} [\![\varphi]\!]\rho[n/k]
\end{array}$$

Figure 8 Interpretation $[\![\varphi]\!]\rho \in \mathcal{P}([\![\tau]\!])$ of $\varphi \in \mathcal{L}(\Sigma;\tau)$, where $\triangle \in \{\pi_1, \pi_2, \mathsf{in}_1, \mathsf{in}_2, \mathsf{fold}\}$.

Let $(X, \leq)$ be a dcpo. A set $U \subseteq X$ is *Scott-open*, notation $U \in \mathcal{O}(X)$, if U is *saturated* (if $x \in U$ and $x \leq y$ in X, then $y \in U$), and if moreover U is inaccessible by directed sups, in the sense that if $\bigvee D \in U$ with $D \subseteq X$ directed, then $D \cap U \neq \emptyset$. This equips X with a topology, called the *Scott topology*. A function between dcpos is Scott-continuous precisely when it is continuous for the Scott topology. See e.g. [4, §1.2] or [3, §2.3]. The following is well-known (see e.g. [3, Theorem 7.2.29] or [12, Theorem 7.47]).

▶ **Theorem 5.3.** *The Scott topology of a Scott domain is spectral.*

There are different equivalent definitions of spectral spaces, see e.g. [11, 1.1.5], [12, §6.1] or [13, 9.5.1]. For now, a *spectral* space is T_0, well-filtered, and the compact-opens are stable under finite intersections and form a basis of the topology. See [31, §D] for details.

Scott topologies are always T_0 since $x \leq y$ in a dcpo X if, and only if, $x \in U$ implies $y \in U$ for each Scott-open U. Given a finite $d \in X$, it is easily seen that $\uparrow d = \{x \in X \mid d \leq x\}$ is *compact-open* (i.e. compact and Scott-open). In fact, if X is algebraic then the Scott-opens are exactly the unions of $\uparrow d$'s, with d finite in X. Each cpo X is trivially compact since $\uparrow \bot = X$. More importantly, if X is a Scott domain then $\uparrow d \cap \uparrow d'$ is compact for all finite $d, d' \in X$ (by bounded-completeness, if $\uparrow d \cap \uparrow d'$ is non-empty, then $d \vee d'$ is defined, finite and such that $\uparrow(d \vee d') = \uparrow d \cap \uparrow d'$). It follows that the set $\mathcal{KO}(X)$ of compact-open subsets of X is stable under finite intersections and forms a basis of the Scott topology.

Proposition 5.4 below states that Scott domains are *well-filtered*. This will yield the topological classification of polarised formulae in Proposition 5.8 (§ 5.3). We let $\mathcal{KS}(X)$ be the set of *compact-saturated* (i.e. compact and saturated) subsets of X. Note that a set $S \subseteq X$ is saturated if, and only if, $S = \bigcap\{U \in \mathcal{O}(X) \mid S \subseteq U\}$. Proposition 5.4 implies that $Q \in \mathcal{KS}(X)$ if, and only if, $Q = \bigcap\{K \in \mathcal{KO}(X) \mid Q \subseteq K\}$, and thus that $\mathcal{KS}(X)$ is stable under all intersections. A subset $\mathcal{Q}$ of a poset P is *codirected* if $\mathcal{Q}$ is directed in P^{op}.

▶ **Proposition 5.4** (Well-Filteredness). *Each Scott domain X is* well-filtered*: given a codirected set $\mathcal{Q} \subseteq \mathcal{KS}(X)$, if $\bigcap \mathcal{Q} \subseteq U$ with U Scott-open, then $Q \subseteq U$ for some $Q \in \mathcal{Q}$.*

▶ **Corollary 5.5.** *If X is a Scott-domain, then $\mathcal{KS}(X)$ is stable under all intersections.*

▶ Remark 5.6 (De Groot Duality). In a spectral space X, the compact-saturated sets are exactly the intersections of compact-opens, while the opens are exactly the unions of compact-opens. This symmetry between $\mathcal{KS}(X)$ and $\mathcal{O}(X)$ is formalised by *de Groot* duality. A nice panorama from the more general point of view of *stably compact spaces* is given in [24].[1] ⌟

5.3 Semantics of Formulae

A *valuation* of a fixpoint context Σ is a function ρ taking fixpoint variables p with $(p\colon \tau) \in \Sigma$ to sets $\rho(p) \in \mathcal{P}([\![\tau]\!])$, and iteration variables k to natural numbers $\rho(k) \in \mathbb{N}$. A valuation ρ thus interprets iteration terms t as natural numbers $[\![t]\!]\rho \in \mathbb{N}$. Figure 8 defines the

[1] In a stably compact space X, $\mathcal{KO}(X)$ may not be a basis. A stably compact algebraic dcpo is spectral.

$$
\begin{array}{rcll}
S \in \mathcal{P}([\![\tau_i]\!]) & \longmapsto & [\![[\pi_i]]\!](S) & \coloneqq \{x \in [\![\tau_1 \times \tau_2]\!] \mid [\![\pi_i]\!](x) \in S\} \\
S \in \mathcal{P}([\![\tau_i]\!]) & \longmapsto & [\![[\mathsf{in}_i]]\!](S) & \coloneqq \{[\![\mathsf{in}_i]\!](x) \in [\![\tau_1 + \tau_2]\!] \mid x \in S\} \\
S \in \mathcal{P}([\![\tau[\mathsf{rec}\,\alpha.\tau/\alpha]]\!]) & \longmapsto & [\![[\mathsf{fold}]]\!](S) & \coloneqq \{x \in [\![\mathsf{rec}\,\alpha.\tau]\!] \mid [\![\mathsf{unfold}]\!](x) \in S\} \\
S \in \mathcal{P}([\![\sigma]\!])\,,\, T \in \mathcal{P}([\![\tau]\!]) & \longmapsto & (S \Vdash T) & \coloneqq \{f \in [\![\sigma \to \tau]\!] \mid \forall x \in S,\ f(x) \in T\}
\end{array}
$$

Figure 9 Semantic modalities, where $i = 1, 2$.

interpretation $[\![\varphi]\!]\rho \in \mathcal{P}([\![\tau]\!])$ of a formula $\varphi \in \mathcal{L}(\Sigma;\tau)$ under a valuation ρ of Σ, where $(p.\varphi)^n(\mathsf{False})$ and $(p.\varphi)^n(\mathsf{True})$ are defined in Equation (2), § 3.1, and where the semantic modalities $[\![[\triangle]]\!]$ and $(-) \Vdash (-)$ are defined in Figure 9. See [31, §E] for details.

In view of Figure 9, it is clear that $[\![\mathsf{in}_1]\!](S_1) \cap [\![\mathsf{in}_2]\!](S_2)$ is always empty. This yields the soundness of the rule (I) in Figure 4b (cf Remark 3.7). The semantic realizability implication $(-) \Vdash (-)$ is clearly contravariant in its first argument and covariant in its second argument. Besides, the polarity constraints in Equation (1) (§ 3.1), as well as the rule (WF) in Figure 4e, come from the following crucial consequence of well-filteredness (Proposition 5.4).

▶ **Lemma 5.7.** *Given $V \in \mathcal{O}([\![\sigma]\!])$ and a codirected $\mathcal{Q} \subseteq \mathcal{KS}([\![\tau]\!])$, if $f \in \bigcap \mathcal{Q} \Vdash V$, then there is some $Q \in \mathcal{Q}$ such that $f \in Q \Vdash V$ already. Moreover, given $Q \in \mathcal{KS}([\![\tau]\!])$ we have $Q \Vdash V \in \mathcal{O}([\![\tau \to \sigma]\!])$ and $V \Vdash Q \in \mathcal{KS}([\![\sigma \to \tau]\!])$.*

▶ **Proposition 5.8.** *If $\varphi \in \mathcal{L}^+(\tau)$ then $[\![\varphi]\!]\rho \in \mathcal{O}([\![\tau]\!])$. If $\varphi \in \mathcal{L}^-(\tau)$ then $[\![\varphi]\!]\rho \in \mathcal{KS}([\![\tau]\!])$. In particular, if $\varphi \in \mathcal{L}^{\pm}(\tau)$ then $[\![\varphi]\!]\rho \in \mathcal{KO}([\![\tau]\!])$.*

Our polarised logical languages $\mathcal{L}^+$ and $\mathcal{L}^-$ are motivated by the topological classification in Proposition 5.8. In view of Figure 2d, the following Lemma 5.9 implies that the unions and intersections interpreting $(\exists k)\varphi$ and $(\forall k)\varphi$ are directed and codirected, respectively. It also yields the soundness of the rules ($\exists$M) and ($\forall$M) in Figure 4c.

▶ **Lemma 5.9.** *If k Pos φ (resp. k Neg φ) then the function $n \mapsto [\![\varphi]\!]\rho[n/k]$ is monotone (resp. antimonotone), that is, $n \leq m$ (resp. $m \leq n$) implies $[\![\varphi]\!]\rho[n/k] \subseteq [\![\varphi]\!]\rho[m/k]$.*

Formulae satisfy an important Scott-(co)continuity property w.r.t. their fixpoint variables. Given $\varphi \in \mathcal{L}(\Sigma;\tau)$ and a valuation ρ of Σ, for each $(p\colon \sigma) \in \Sigma$ we have a function

$$[\![p.\varphi]\!]\rho\colon \mathcal{P}([\![\sigma]\!]) \longrightarrow \mathcal{P}([\![\tau]\!]), \quad S \longmapsto [\![\varphi]\!]\rho[S/p]$$

Note that in the case of $\varphi \in \mathcal{L}(\Sigma, p\colon \tau;\tau)$, for all $\psi \in \mathcal{L}(\Sigma;\tau)$ and all $n \in \mathbb{N}$, we have $([\![p.\varphi]\!]\rho)^n([\![\psi]\!]\rho) = [\![(p.\varphi)^n(\psi)]\!]\rho$, where $(p.\varphi)^n(\psi)$ is the formula in Equation (2).

▶ **Proposition 5.10.** *Let ρ be a valuation of Σ, and let $(p\colon \sigma) \in \Sigma$.*
(1) *If $\varphi \in \mathcal{L}^s(\Sigma;\tau)$, then $[\![p.\varphi]\!]\rho$ is monotone.*
(2) *If $\varphi \in \mathcal{L}^+(\Sigma;\tau)$, then $[\![p.\varphi]\!]\rho$ is Scott-continuous (i.e. preserves directed unions).*
(3) *If $\varphi \in \mathcal{L}^-(\Sigma;\tau)$, then $[\![p.\varphi]\!]\rho$ is Scott-cocontinuous (i.e. preserves codirected intersections).*

▶ **Example 5.11.** Let $\varphi \in \mathcal{L}^+(\Sigma, p\colon \tau;\tau)$ and $\psi \in \mathcal{L}^-(\Sigma, p\colon \tau;\tau)$. It follows from Proposition 5.10 that the least fixpoint of $[\![p.\varphi]\!]\rho$ and the greatest fixpoint of $[\![p.\psi]\!]\rho$ are

$$[\![(\exists k)(\mu^k p)\varphi]\!]\rho = \bigcup_{n\in\mathbb{N}} ([\![p.\varphi]\!]\rho)^n(\mathsf{False}) \quad \text{and} \quad [\![(\forall \ell)(\nu^\ell p)\psi]\!]\rho = \bigcap_{n\in\mathbb{N}} ([\![p.\psi]\!]\rho)^n(\mathsf{True})$$

On streams, we get that $\Diamond\varphi$ defines an *open* set, while $\Box\psi$ defines a *compact-saturated* set (with φ positive and ψ negative). For instance, $[\![\Box[\mathsf{hd}][\mathsf{tt}]]\!] = \{[\![\mathsf{tt}]\!]^\omega\}$ is compact-saturated in $[\![\mathsf{Stream\ Bool}]\!] = [\![\mathsf{Bool}]\!]^\omega$. This contrasts with the case of ω-words, for which $\{\mathsf{tt}^\omega\}$ is a *closed*

subset of $\{\mathsf{tt}, \mathsf{ff}\}^\omega$ for the product topology.[2] Besides, ω-words over a finite alphabet form a Stone (Boolean) space, i.e. a spectral space in which the compact-saturated sets coincide with the closed ones. But in the case of the Scott domain of streams, proper compact-saturated sets (such as $\{[\![\mathsf{tt}]\!]^\omega\}$) are never closed. We think this makes the case of de Groot duality (Remark 5.6) for temporal properties on streams.

In the context of Example 3.3, the significance of well-filteredness (Proposition 5.4) in Lemma 5.7 is that a Scott-continuous function $f\colon [\![\mathsf{Tree\,Bool} \to \mathsf{Stream\,Nat}]\!]$, say $[\![\mathsf{count} \circ \mathsf{bft}]\!]$, satisfies the formula in Equation (4) *if, and only if,* there is some $m \in \mathbb{N}$ such that for each totally defined $t \in [\![\mathsf{Tree\,Bool}]\!]$, the length-$m$ prefix of $f(t)$ contains a totally defined number $\geq n$ provided that in the depth-m truncation of t, the children of the root have a maximal path consisting of tt's and one consisting of ff's. Hence, only a depth-m exploration of $f(t)$ is needed to establish the property, where $m \in \mathbb{N}$ depends on f but *not* on t. ⌟

Proposition 5.10 yields the rules $(\iota/\exists)$ and $(\forall/\iota)$ in Figure 4c. The following Proposition 5.12 is crucial for the rules $(\Vdash\!\!\to/\vee)$ and (CC) in Figure 4e, as well as for completeness.

▶ **Proposition 5.12.** *Given $\delta \in \mathcal{L}^\wedge(\tau)$, if $[\![\delta]\!] \neq \emptyset$ then $[\![\delta]\!] = {\uparrow}d$ for some finite $d \in [\![\tau]\!]$. Conversely, if $d \in [\![\tau]\!]$ is finite, then ${\uparrow}d = [\![\delta]\!]$ for some $\delta \in \mathcal{L}^\wedge(\tau)$.*

▶ **Theorem 5.13** (Soundness of Deduction)**.**
(1) *For all $\delta \in \mathcal{L}^\wedge(\tau)$, if $\mathcal{C}(\delta)$ is derivable, then $[\![\delta]\!] \neq \emptyset$.*
(2) *For all $\varphi, \psi \in \mathcal{L}^s(\Sigma; \tau)$, if $\psi \vdash \varphi$ is derivable, then $[\![\psi]\!]\rho \subseteq [\![\varphi]\!]\rho$.*

The following is the key logical fact toward completeness. It also entails the completeness of the neutral fragment (which we actually do not need for our main Theorem 5.17).

▶ **Theorem 5.14** (Completeness of $\mathcal{L}^\wedge$)**.** *Let $\varphi, \psi \in \mathcal{L}^\wedge(\tau)$.*
(1) *If $[\![\varphi]\!] \neq \emptyset$, then $\mathcal{C}(\varphi)$ is derivable.*
(2) *If $[\![\psi]\!] \subseteq [\![\varphi]\!]$, then $\psi \vdash \varphi$ is derivable.*

5.4 Semantics of the Refinement Type System

The interpretation $[\![T]\!] \subseteq [\![|T|]\!]$ of a refinement type T is defined as $[\![\{\tau \mid \varphi\}]\!] \coloneqq [\![\varphi]\!]$ and $[\![U \to T]\!] \coloneqq [\![U]\!] \Vdash\!\!\to [\![T]\!]$ and $[\![T \times U]\!] \coloneqq [\![[\pi_1]]\!]([\![T]\!]) \cap [\![[\pi_2]]\!]([\![U]\!])$.

▶ **Definition 5.15** (Sound Judgement)**.** *A judgement $\mathcal{E} \vdash M\colon T$ with $\mathcal{E} = x_1\colon U_1, \ldots, x_n\colon U_n$ is* sound *if $|\mathcal{E}| \vdash M\colon |T|$ is derivable and if moreover $[\![M]\!](u_1, \ldots, u_n) \in [\![T]\!]$ whenever $u_i \in [\![U_i]\!]$ for all $i = 1, \ldots, n$.*

▶ **Theorem 5.16** (Soundness of Typing)**.** *If $\mathcal{E} \vdash M\colon T$ is derivable then it is sound.*

It follows from Examples 5.2 and 5.11 that $\vdash \mathsf{count} \circ \mathsf{bft}\colon T^+$ is sound, where T^+ is either type in Example 4.1. This judgement is also derivable, thanks to our main result:

▶ **Theorem 5.17** (Positive Completeness)**.** *If $\mathcal{E}^- \vdash M\colon T^+$ is sound, then it is derivable.*

Write $T^\wedge$ for a refinement type which only contains formulae in $\mathcal{L}^\wedge$, and similarly for $\mathcal{E}^\wedge$. We obtain Theorem 5.17 by reduction to judgements of the form $\mathcal{E}^\wedge \vdash M\colon T^\wedge$, for which completeness essentially amounts to [2] (though formally our system differs).

▶ **Theorem 5.18.** *If $\mathcal{E}^\wedge \vdash M\colon T^\wedge$ is sound, then it is derivable.*

[2] See [33] for a comparison of the Scott topology on streams with the product topology on ω-words.

The reduction of Theorem 5.17 to Theorem 5.18 is an adaptation of [7, 32]. Consider first the case of a sound $\vdash M : T$, where T is positive. By Lemma 4.3 and Proposition 3.9, let $\varphi \in \mathcal{L}^{\pm}(|T|)$ such that $T \simeq \{|T| \mid (\exists k)\varphi\}$. Since the judgement $\vdash M : T$ is sound, there is some $m \in \mathbb{N}$ such that $\vdash M : \{|T| \mid \varphi[m/k]\}$ is also sound. By Lemmas 3.8 and 3.6, the closed formula $\varphi[m/k] \in \mathcal{L}^{\pm}(|T|)$ is equivalent to a disjunction of formulae $\gamma \in \mathcal{L}^{\wedge}(|T|)$. But there must be at least one of these γ's such that the judgement $\vdash M : \{|T| \mid \gamma\}$ is sound, and we can apply Theorem 5.18. Since $\{|T| \mid \gamma\} \preceq \{|T| \mid \varphi[m/k]\} \preceq \{|T| \mid (\exists k)\varphi\}$, we obtain a derivation of $\vdash M : \{|T| \mid (\exists k)\varphi\}$ from a derivation of $\vdash M : \{|T| \mid \gamma\}$.

Consider now the case of $x : U \vdash M : T$, where U is negative. Let $\psi \in \mathcal{L}^{\pm}(|U|)$ such that $U \simeq \{|U| \mid (\forall \ell)\psi\}$. Since $[\![T]\!]$ is Scott-open and $[\![M]\!]$ is Scott-continuous, well-filteredness (Proposition 5.4) yields some $m \in \mathbb{N}$ such that $[\![\psi[m/\ell]]\!] \subseteq [\![M]\!]^{-1}([\![T]\!])$, i.e. such that $x : \{|U| \mid \psi[m/\ell]\} \vdash M : T$ is sound. We can similarly write $\psi[m/\ell]$ as a disjunction of $\delta \in \mathcal{L}^{\wedge}(|U|)$, and each disjunct δ leads to a sound judgement $x : \{|U| \mid \delta\} \vdash M : T$. If $\delta \not\vdash \mathsf{False}$, then Proposition 5.12 gives a finite $d \in [\![|U|]\!]$ such that $[\![\delta]\!] = \uparrow d$. Since $[\![T]\!]$ is saturated, we have $\uparrow d \subseteq [\![M]\!]^{-1}([\![T]\!])$ if, and only if, $[\![M]\!](d) \in [\![T]\!]$. Unfolding T similarly as above, the condition $[\![M]\!](d) \in [\![T]\!]$ yields some $\gamma \in \mathcal{L}^{\wedge}(|T|)$ with $\{|T| \mid \gamma\} \preceq T$ and such that $x : \{|U| \mid \delta\} \vdash M : \{|T| \mid \gamma\}$ is sound. See [31, §F] for details.

▶ Remark 5.19 (Undecidability). Positive judgements are semi-decidable but not decidable: by Turing-completeness, there is a $K : \mathsf{Nat} \to \mathsf{Nat}$ such that $K\mathsf{n}$ evaluates the nth partial recursive function on input n. Hence the total function which given $n \in \mathbb{N}$ decides the derivability (or soundness) of $\vdash K\mathsf{n} : \{\mathsf{Nat} \mid [\mathsf{tot}]\}$, is not recursive ([29, Theorem II.2.3]). ⌟

6 Conclusion

We presented a finitary refinement type system which is complete for a set of Scott-open specifications strictly extending [1, 2]. We moreover noted that these specifications are not decidable for the typed λ-calculus under consideration (Remark 5.19 above).

An important direction of future work is to characterise the expressiveness of our logic. Example 5.11 means that our positive (resp. negative) fragment corresponds to a modal μ-calculus on (finitary) polynomial types, restricted to negation-free formulae with least (resp. greatest) fixpoints. However, we cannot yet be as precise for higher-order types (with formulae involving the realizability implication $\Vdash\!\!\rightarrow$).

An other direction for future work is to extend this system to liveness properties (as in Remark 3.4), while still targeting a complete finitary type system (in contrast with [19, 32]).

Our system has sum types (so as to have a recursive type of natural numbers), but the category **Scott** does not have finite coproducts. Working with Call-By-Push-Value (CBPV) [25, 26], for the usual adjunction between dcpos and cpos with strict functions, may lead to a cleaner situation. In the long run, it would be nice if this basis could extend to enriched models of CBPV, so as to handle further computational effects. Print and global store are particularly relevant, as an important trend in proving temporal properties considers programs generating streams of events. Major works in this line include [35, 15, 16, 27, 23, 38, 27, 34]. In contrast with ours, these approaches are based on trace semantics of syntactic expressions rather than denotational domains.

In a different direction, we think the approach of this paper could extend to linear types [17, 28, 41], in particular for the λ-calculus HOPLA, and possibly relying on the categorical study of [9].

References

1. S. Abramsky. Domain theory in logical form. In *Proceedings of the Second Annual IEEE Symposium on Logic in Computer Science (LICS 1987)*, pages 47–53. IEEE Computer Society Press, June 1987.
2. S. Abramsky. Domain Theory in Logical Form. *Ann. Pure Appl. Log.*, 51(1-2):1–77, 1991. doi:10.1016/0168-0072(91)90065-T.
3. S. Abramsky and A. Jung. Domain Theory. In S. Abramsky, D.M. Gabbay, and Maibaum T.S.E., editors, *Handbook of Logic in Computer Science*, chapter 1. Clarendon Press, Oxford, 1995.
4. R. M. Amadio and P.-L. Curien. *Domains and Lambda-Calculi*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.
5. C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
6. U. Berger, R. Matthes, and A. Setzer. Martin Hofmann's Case for Non-Strictly Positive Data Types. In P. Dybjer, J. Espírito Santo, and L. Pinto, editors, *Proceedings of TYPES 2018*, volume 130 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.TYPES.2018.1.
7. M. M. Bonsangue and J. N. Kok. Infinite intersection types. *Information and Computation*, 186(2):285–318, 2003. doi:10.1016/S0890-5401(03)00143-3.
8. J. Bradfield and C. Stirling. Modal Mu-Calculi. In P. Blackburn, J. Van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 721–756. Elsevier, 2007. doi:10.1016/S1570-2464(07)80015-2.
9. M. Bunge and J. Funk. *Singular Coverings of Toposes*, volume 1890 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 2006.
10. T. Coquand and G.-Q. Zhang. Sequents, Frames, and Completeness. In *Proceedings of the 14th Annual Conference of the EACSL on Computer Science Logic*, pages 277–291, London, UK, UK, 2000. Springer-Verlag.
11. M. Dickmann, N. Schwartz, and M. Tressl. *Spectral Spaces*. New Mathematical Monographs. Cambridge University Press, Cambirdge, 2019. doi:10.1017/9781316543870.
12. M. Gehrke and S. van Gool. *Topological Duality for Distributive Lattices: Theory and Applications*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 2024.
13. J. Goubault-Larrecq. *Non-Hausdorff Topology and Domain Theory: Selected Topics in Point-Set Topology*. New Mathematical Monographs. Cambridge University Press, Cambridge, 2013. doi:10.1017/CBO9781139524438.
14. I. Hodkinson and M. Reynolds. Temporal Logic. In P. Blackburn, J. Van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 655–720. Elsevier, 2007. doi:10.1016/S1570-2464(07)80014-0.
15. M. Hofmann and W. Chen. Abstract interpretation from Büchi automata. In T. A. Henzinger and D. Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 51:1–51:10. ACM, 2014. doi:10.1145/2603088.2603127.
16. M. Hofmann and J. Ledent. A cartesian-closed category for higher-order model checking. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005120.
17. M. Huth, A. Jung, and K. Keimel. Linear types and approximation. *Mathematical Structures in Computer Science*, 10(6):719–745, 2000. URL: http://journals.cambridge.org/action/displayAbstract?aid=71049.
18. H. Huwig and A. Poigné. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73(1):101–112, 1990. doi:10.1016/0304-3975(90)90165-E.

19 G. Jaber and C. Riba. Temporal Refinements for Guarded Recursive Types. In N. Yoshida, editor, *Proceedins of ESOP'21*, volume 12648 of *Lecture Notes in Computer Science*, pages 548–578. Springer, 2021. doi:10.1007/978-3-030-72019-3_20.

20 P.T. Johnstone. *Stone Spaces*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1982.

21 D. O. Kidney and N. Wu. Hyperfunctions: Communicating Continuations. In *Proceedings of the 53rd ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2026)*. ACM, 2026. doi:10.1145/3776649.

22 N. Kobayashi, N. Tabuchi, and H. Unno. Higher-Order Multi-Parameter Tree Transducers and Recursion Schemes for Program Verification. In *POPL '10: Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 495–508, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1707801.1706355.

23 E. Koskinen and T. Terauchi. Local Temporal Reasoning. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS'14, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2603088.2603138.

24 J. Lawson. Stably compact spaces. *Mathematical Structures in Computer Science*, 21(1):125–169, 2011. doi:10.1017/S0960129510000319.

25 P. B. Levy. *Call-By-Push-Value*. Semantics Structures in Computation. Springer, Dordrecht, 2003. doi:10.1007/978-94-007-0954-6.

26 P. B. Levy. Call-by-Push-Value. *ACM SIGLOG News*, 9(2):7–29, May 2022. doi:10.1145/3537668.3537670.

27 Y. Nanjo, H. Unno, E. Koskinen, and T. Terauchi. A Fixpoint Logic and Dependent Effects for Temporal Property Verification. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS'18, pages 759–768, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209204.

28 M. Nygaard and G. Winskel. Full Abstraction for HOPLA. In R.M. Amadio and D. Lugiez, editors, *Proceedings of CONCUR 2003*, volume 2761 of *Lecture Notes in Computer Science*, pages 378–392. Springer, 2003. doi:10.1007/978-3-540-45187-7_25.

29 P. Odifreddi. *Classical Recursion Theory*, volume 143 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, second edition, 1999.

30 B. C. Pierce. *Types and Programming Languages*. The MIT Press, 1st edition, 2002.

31 C. Riba and A. Donadille. A Complete Finitary Refinement Type System for Scott-Open Properties. Full version, available on arXiv, January 2026. doi:10.48550/arXiv.2601.23082.

32 C. Riba and A. Kejikian. Infinitary Refinement Types for Temporal Properties in Scott Domains. In *Proceedings of WoLLIC'25*, 2025. Full version available on arXiv (https://arxiv.org/abs/2502.11917). doi:10.1007/978-3-031-99536-1_11.

33 C. Riba and S. Stern. Liveness Properties in Geometric Logic for Domain-Theretic Streams. In *Proceedings of JFLA'24*, 2024. Full version available on arXiv (https://arxiv.org/abs/2310.12763). URL: https://inria.hal.science/hal-04407194.

34 T. Sekiyama and H. Unno. Temporal Verification with Answer-Effect Modification: Dependent Temporal Type-and-Effect System with Delimited Continuations. *Proc. ACM Program. Lang.*, 7(POPL), January 2023. Full version available on arXiv at https://arxiv.org/abs/2207.10386.

35 C. Skalka, S. Smith, and D. Van Horn. Types and Trace Effects of Higher Order Programs. *J. Funct. Program.*, 18(2):179–249, March 2008. doi:10.1017/S0956796807006466.

36 C. Sprenger and M. Dam. On the Structure of Inductive Reasoning: Circular and Tree-Shaped Proofs in the μ-Calculus. In A. D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, volume 2620 of *Lecture Notes in Computer Science*, pages 425–440. Springer, 2003. doi:10.1007/3-540-36576-1_27.

37 T. Streicher. *Domain-Theoretic Foundations of Functional Programming.* World Scientific, 2006. doi:10.1142/6284.

38 H. Unno, Y. Satake, and T. Terauchi. Relatively complete refinement type system for verification of higher-order non-deterministic programs. *Proc. ACM Program. Lang.*, 2(POPL):12:1–12:29, 2018. doi:10.1145/3158100.

39 S. Vickers. *Topology via Logic.* Cambridge University Press, USA, 1989.

40 S. Vickers. Locales and Toposes as Spaces. In M. Aiello, I. Pratt-Hartman, and J. van Benthem, editors, *Handbook of Spatial Logics*, chapter 8, pages 429–496. Springer, 2007. doi:10.1007/978-1-4020-5587-4_8.

41 G. Winskel. Linearity and nonlinearity in distributed computation. In T. Ehrhard, J.-Y. Girard, and P. Ruet, editors, *Linear Logic in Computer Science*, London Mathematical Society Lecture Note Series. Cambridge University Press, 2004.

42 G. Zhang. *Logic of Domains.* Progress in Theoretical Computer Science. Birkhäuser, Boston, MA, 1991. doi:10.1007/978-1-4612-0445-9.