Temporal Refinements for Guarded Recursive Types
(full version)
Guilhem Jaber, Colin Riba

To cite this version:
hal-02512655v5

HAL Id: hal-02512655
https://hal.archives-ouvertes.fr/hal-02512655v5
Preprint submitted on 14 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Temporal Refinements for Guarded Recursive Types

Guilhem Jaber\(^1\) and Colin Riba\(^2\)

Université de Nantes, LS2N CNRS, Inria, France guilhem.jaber@univ-nantes.fr
Univ Lyon, EnsL, UCBL, CNRS, LIP, F-69342, LYON Cedex 07, France colin.riba@ens-lyon.fr

Abstract. We propose a logic for temporal properties of higher-order programs that handle infinite objects like streams or infinite trees, represented via coinductive types. Specifications of programs use safety and liveness properties. Programs can then be proven to satisfy their specification in a compositional way, our logic being based on a type system. The logic is presented as a refinement type system over the guarded \(\lambda\)-calculus, a \(\lambda\)-calculus with guarded recursive types. The refinements are formulae of a modal \(\mu\)-calculus which embeds usual temporal modal logics such as LTL and CTL. The semantics of our system is given within a rich structure, the topos of trees, in which we build a realizability model of the temporal refinement type system. We use in a crucial way the connection with set-theoretic semantics to handle liveness properties.

Keywords: coinductive types, guarded recursive types, \(\mu\)-calculus, refinement types, topos of trees.

1 Introduction

Functional programming is by now well established to handle infinite data, thanks to declarative definitions and equational reasoning on high-level abstractions, in particular when infinite objects are represented with coinductive types. In such settings, programs in general do not terminate, but are expected to compute a part of their output in finite time. For example, a program expected to generate a stream should produce the next element in finite time: it is productive.

Our goal is to prove input-output temporal properties of higher-order programs that handle coinductive types. Logics like LTL, CTL or the modal \(\mu\)-calculus are widely used to formulate, on infinite objects, safety and liveness properties. Safety properties state that some “bad” event will not occur, while liveness properties specify that “something good” will happen (see e.g. [9]). Typically, modalities like \(\square\) (always) or \(\Diamond\) (eventually) are used to write properties of streams or infinite trees and specifications of programs over such data.

We consider temporal refinement types \(\{A \mid \varphi\}\), where \(A\) is a standard type of our programming language, and \(\varphi\) is a formula of the modal \(\mu\)-calculus. Using refinement types [24], temporal connectives are not reflected in the programming.
language, and programs are formally independent from the shape of their temporal specifications. One can thus give different refinement types to the same program. For example, the following two types can be given to the same map function on streams:

\[
\text{map} : (\{B|\psi\} \rightarrow \{A|\varphi\}) \rightarrow \{\text{Str } B|\Box\Diamond\text{hd}\psi\} \rightarrow \{\text{Str } A|\Box\Diamond\text{hd}\varphi\} \quad \text{(⋆)}
\]

These types mean that given \(f : B \rightarrow A\) s.t. \(f(b)\) satisfies \(\varphi\) if \(b\) satisfies \(\psi\), the function \((\text{map } f)\) takes a stream with infinitely many (resp. ultimately all) elements satisfying \(\psi\) to one with infinitely many (resp. ultimately all) elements satisfying \(\varphi\). For \(\varphi\) a formula over \(A\), \(\text{hd}\varphi\) is a formula over streams of \(A\)'s which holds on a given stream if \(\varphi\) holds on its head element.

It is undecidable whether a given higher-order program satisfies a given input-output temporal property written with formulae of the modal \(\mu\)-calculus [15]. Having a type system is a partial workaround to this obstacle, which moreover enables to reason compositionally on programs, by decomposing a specification to the various components of a program in order to prove its global specification.

Our system is built on top of the guarded \(\lambda\)-calculus [20], a higher-order programming language with guarded recursion [57]. Guarded recursion is a simple device to control and reason about unfoldings of fixpoints. It can represent coinductive types [55] and provides a syntactic compositional productivity check [5].

Safety properties (e.g. \(\Box[\text{hd}]\varphi\)) can be correctly represented with guarded fixpoints, but not liveness properties (e.g. \(\Diamond[\text{hd}]\varphi, \Box\Diamond[\text{hd}]\varphi, \Box\Diamond[\text{hd}]\varphi\)). Combining liveness with guarded recursion is a challenging problem since guarded fixpoints tend to have unique solutions. Existing approaches to handle temporal types in presence of guarded recursion face similar difficulties. Functional reactive programming (FRP) [23] provides a Curry-Howard correspondence for temporal logics [35,36,18] in which logical connectives are reflected as programming constructs. When combining FRP with guarded recursion [48,7], and in particular to handle liveness properties [8], uniqueness of guarded fixpoints is tempered by specific recursors for temporal types.

Our approach is different from [8], as we wish as much as possible the logical level not to impact the program level. We propose a two level system, with the lower or internal level, which interacts with guarded recursion and at which only safety properties are correctly represented, and the higher or external one, at which liveness properties are correctly handled, but without direct access to guarded recursion. By restricting to the alternation-free modal \(\mu\)-calculus, in which fixpoints can always be computed in \(\omega\)-steps, one can syntactically reason on finite unfoldings of liveness properties, thus allowing for crossing down the safety barrier. Soundness is proved by a realizability interpretation based on the semantics of guarded recursion in the topos of trees [13], which correctly represents the usual set-theoretic final coalgebras of polynomial coinductive types [55].

We provide example programs involving linear structures (colists, streams, fair streams [18]) and branching structures (resumptions à la [48]), for which we prove liveness properties similar to (⋆) above. Our system also handles safety properties on breadth-first (infinite) tree traversals à la [39] and [10].
Cons\textsuperscript{\textcircled{S}} := \lambda x.\lambda s. \text{fold}(x, s) : A \rightarrow \text{Str}\textsuperscript{\textcircled{S}} A \rightarrow \text{Str}\textsuperscript{\textcircled{S}} A
\hd\textsuperscript{\textcircled{S}} := \lambda s.\pi_0(\text{unfold } s) : \text{Str}\textsuperscript{\textcircled{S}} A \rightarrow A
\tl\textsuperscript{\textcircled{S}} := \lambda s.\pi_1(\text{unfold } s) : \text{Str}\textsuperscript{\textcircled{S}} A \rightarrow \text{Str}\textsuperscript{\textcircled{S}} A
\text{map}\textsuperscript{\textcircled{S}} := \lambda f.\text{fix}(g).\lambda s.\text{Cons} g(\text{f}(\text{hd}\textsuperscript{\textcircled{S}} s))(g \circ (\text{tl}\textsuperscript{\textcircled{S}} s)) : (B \rightarrow A) \rightarrow \text{Str}\textsuperscript{\textcircled{S}} B \rightarrow \text{Str}\textsuperscript{\textcircled{S}} A

Fig. 1. Constructor, Destructors and Map on Guarded Streams.

Organization of the paper. We give an overview of our approach in §2. Then §3 presents the syntax of the guarded \(\lambda\)-calculus. Our base temporal logic (without liveness) is introduced in §4 and is used to define our refinement type system in §5. Liveness properties are handled in §6. The semantics is given in §7, and §8 presents examples. Finally, we discuss related work in §9 and future work in §10. Table 4 (§8) gathers the main refinement types we can give to example functions, most of them defined in Table 3. Omitted material is available in the Appendices.

2 Outline

Overview of the Guarded \(\lambda\)-Calculus. Guarded recursion enforces productivity of programs using a type system equipped with a type modality \(\triangleright\), in order to indicate that one has access to a value not right now but only “later”. One can define guarded streams \(\text{Str}\textsuperscript{\textcircled{S}} A\) over a type \(A\) via the guarded recursive definition \(\text{Str}\textsuperscript{\textcircled{S}} A = A \times \triangleright \text{Str}\textsuperscript{\textcircled{S}} A\). Streams that inhabit this type have their head available now, but their tail only one step in the future. The type modality \(\triangleright\) is reflected in programs with the \textit{next} operation. One also has a fixpoint constructor on terms \(\text{fix}(x).M\) for guarded recursive definitions. They are typed with

\[
\begin{align*}
E \vdash M : A & \quad E, x : \triangleright A \vdash M : A \\
E \vdash \text{next}(M) : \triangleright A & \quad E \vdash \text{fix}(x).M : A
\end{align*}
\]

This allows for the constructor and basic destructors on guarded streams to be defined as in Fig. 1 where \(\text{fold}(\_\_\_)\) and \(\text{unfold}(\_\_\_)\) are explicit operations for folding and unfolding guarded recursive types. In the following, we use the infix notation \(a ::\textsuperscript{\textcircled{S}} s\) for \(\text{Cons}\textsuperscript{\textcircled{S}} a s\). Using the fact that the type modality \(\triangleright\) is an applicative functor \[54\], we can distribute \(\triangleright\) over the arrow type. This is represented in the programming language by the infix applicative operator \(\circ\). With it, one can define the usual map function on guarded streams as in Fig. 1.

Compositional Safety Reasoning on Streams. Given a property \(\varphi\) on a type \(A\), we would like to consider a subtype of \(\text{Str}\textsuperscript{\textcircled{S}} A\) that selects those streams whose elements all satisfy \(\varphi\). To do so, we use a temporal modal formula \(\square[\text{hd}]\varphi\), and consider the \textit{refinement type} \(\{\text{Str}\textsuperscript{\textcircled{S}} A | \square[\text{hd}]\varphi\}\). Suppose for now that we
can give the following refinement types to the basic stream operations:

\[
\begin{align*}
\text{hd}^\tau &: \{\text{Str}^\tau A \mid \Box[\text{hd}]\varphi\} \rightarrow \{A \mid \varphi\} \\
\text{tl}^\tau &: \{\text{Str}^\tau A \mid \Box[\text{hd}]\varphi\} \rightarrow \{\text{Str}^\tau A \mid \Box[\text{hd}]\varphi\} \\
\text{Cons}^\tau &: \{A \mid \varphi\} \rightarrow \{\text{Str}^\tau A \mid \Box[\text{hd}]\varphi\} \rightarrow \{\text{Str}^\tau A \mid \Box[\text{hd}]\varphi\}
\end{align*}
\]

By using the standard typing rules for \(\lambda\)-abstraction and application, together with the rules to type \(\text{fix}(x).M\) and \(\varepsilon\), we can type the function \(\text{map}^\tau\) as

\[
\text{map}^\tau : (\{B \mid \psi\} \rightarrow \{A \mid \varphi\}) \rightarrow \{\text{Str}^\tau B \mid \Box[\text{hd}]\psi\} \rightarrow \{\text{Str}^\tau A \mid \Box[\text{hd}]\varphi\}
\]

### A Manysorted Temporal Logic

Our logical language, taken with minor adaptations from [33], is manysorted: for each type \(A\) we have formulae of type \(A\) (notation \(\vdash \varphi : A\)), where \(\varphi\) selects inhabitants of \(A\).

We use atomic modalities (\([\pi], [\text{fold}], [\text{next}], \ldots\)) in refinements to navigate between types (see Fig. 5). For instance, a formula \(\varphi\) of type \(A_0\), specifying a property over the inhabitants of \(A_0\), can be lifted to the formula \([\pi_0]\varphi\) of type \(A_0 \times A_1\), which intuitively describes those inhabitants of \(A_0 \times A_1\) whose first component satisfy \(\varphi\). Given a formula \(\varphi\) of type \(A\), one can define its "head lift" \(\text{hd}^\varphi\) of type \(\text{Str}^\varphi A\), that enforces \(\varphi\) to be satisfied on the head of the provided stream. Also, one can define a modality \(\Box\) such that given a formula \(\psi : \text{Str}^\varphi A\), the formula \(\Box\psi : \text{Str}^\varphi A\) enforces \(\psi\) to be satisfied on the tail of the provided stream. These modalities are obtained resp. as \([\text{hd}]\varphi := [\text{fold}][\pi_0]\varphi\) and \(\Box\varphi := [\text{fold}][\pi_1][\text{next}]\varphi\). We similarly have atomic modalities \([\text{in}0], [\text{in}1]\) on sum types. For instance, on the type of guarded colists defined as \(\text{CoList}^\tau A := \text{Fix}(X). 1 + A \times \Box X\), we can express the fact that a colist is empty (resp. non-empty) with the formula \([\text{nil}] := [\text{fold}][\text{in}0]\top\) (resp. \([\neg\text{nil}] := [\text{fold}][\text{in}1]\top\).

We also provide a deduction system \(\vdash^\tau \varphi\) on temporal modal formulae. This deduction system is used to define a subtyping relation \(T \leq U\) between refinement types, with \(\{A \mid \varphi\} \leq \{A \mid \psi\}\) when \(\vdash^\tau \varphi \Rightarrow \psi\). The subtyping relation thus incorporates logical reasoning in the type system.

In addition, we have greatest fixpoints formulae \(\nu\alpha \varphi\) (so that formulae can have free typed propositional variables), equipped with Kozen’s reasoning principles [47]. In particular, we can form an always modality as \(\Box\varphi := \nu\alpha. \varphi \land \Box\alpha\), with \(\Box\varphi : \text{Str}^\varphi A\) if \(\varphi : \text{Str}^\varphi A\). The formula \(\Box\varphi\) holds on a stream \(s = (s_i \mid i \geq 0)\), iff \(\varphi\) holds on every substream \((s_i \mid i \geq n)\) for \(n \geq 0\). If we rather start with \(\psi : A\), one first need to lift it to \([\text{hd}]\psi : \text{Str}^\varphi A\). Then \([\Box[\text{hd}]\psi\) means that all the elements of the stream satisfies \(\psi\), since all its suffixes satisfy \([\text{hd}]\psi\).

Table 1 summarizes the different judgments used in this paper.

<table>
<thead>
<tr>
<th>Typing Formulae</th>
<th>Provable</th>
<th>Refinement Types</th>
<th>Subtyping</th>
<th>Typing</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Sigma \vdash \varphi : A)</td>
<td>(\vdash^\tau \varphi)</td>
<td>({A \mid \varphi})</td>
<td>(T \leq U)</td>
<td>(M : T)</td>
</tr>
<tr>
<td>(where (\varphi : A), (\varphi))</td>
<td>(where (\varphi : A), (\varphi))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Syntactic Classes and Judgments.
Beyond Safety. In order to handle liveness properties, we also need to have least fixpoints formulae \( \mu \alpha \varphi \). For example, this would give the eventually modality \( \Box \varphi := \mu \alpha. \varphi \lor \bigcirc \alpha \). With Kozen-style rules, one could then give the following two types to the guarded stream constructor:

\[
\begin{align*}
\text{Cons}^\times : \{ A | \varphi \} & \rightarrow \text{Str}^\times A \rightarrow \{ \text{Str}^\times A | \Box [\text{hd}] \varphi \} \\
\text{Cons}^\times : A & \rightarrow \{ \text{Str}^\times A | \Box [\text{hd}] \varphi \} \rightarrow \{ \text{Str}^\times A | \Box [\text{hd}] \varphi \}
\end{align*}
\]

But consider a finite base type \( B \) with two distinguished elements \( a, b \), and suppose that we have access to a modality \( [b] \) on \( B \) so that terms inhabiting \( \{ B | [b] \} \) must be equal to \( b \). Using the above types for \( \text{Cons}^\times \), we could type the stream with constant value \( a \), defined as \( \text{fix}(s).a := s \), with the type \( \{ \text{Str}^\times B | \Box [\text{hd}][b] \} \) that is supposed to enforce the existence of an occurrence of \( b \) in the stream. Similarly, on colists we would have \( \text{fix}(s).a := s \) of type \( \{ \text{Colist}^\times B | \Box [\text{nil}] \} \), while \( \Box [\text{nil}] \) expresses that a colist will eventually contain a nil, and is thus finite. Hence, liveness properties may interact quite badly with guarded recursion. Let us look at this in a semantic model of guarded recursion.

**Internal Semantics in the Topos of Trees.** The types of the guarded \( \lambda \)-calculus can be interpreted as sequences of sets \( (X(n))_{n > 0} \) where \( X(n) \) represents the values available “at time \( n \)”. In order to interpret guarded recursion, one also needs to have access to functions \( r^X_n : X(n+1) \rightarrow X(n) \), which tell how values “at \( n+1 \)” can be restricted (actually most often truncated) to values “at \( n \)”. This means that the objects used to represent types are in fact presheaves over the poset \( (\mathbb{N} \setminus \{0\}, \leq) \). The category \( S \) of such presheaves is the topos of trees [13]. For instance, the type \( \text{Str}^\times B \) of guarded streams over a finite base type \( B \) is interpreted in \( S \) as \( (B^n)_{n > 0} \), with restriction maps taking \( (b_0, \ldots, b_{n-1}, b_n) \) to \( (b_0, \ldots, b_{n-1}) \). We write \( [A] \) for the interpretation of a type \( A \) in \( S \).

**The Necessity of an External Semantics.** The topos of trees cannot correctly handle liveness properties. For instance, the formula \( \Box [\text{hd}][b] \) cannot describe in \( S \) the set of streams that contain at least one occurrence of \( b \). Indeed, the interpretation of \( \Box [\text{hd}][b] \) in \( S \) is a sequence \( (C(n))_{n > 0} \) with \( C(n) \subseteq B^n \). But any element of \( B^n \) can be extended to a stream which contains an occurrence of \( b \). Hence \( C(n) \) should be equal to \( B^n \), and the interpretation of \( \Box [\text{hd}][b] \) is the whole \( [\text{Str}^\times B] \). More generally, guarded fixpoints have unique solutions in the topos of trees [13], and \( \Box \varphi = \mu \alpha. \varphi \lor \bigcirc \varphi \) gets the same interpretation as \( \nu \alpha. \varphi \lor \Box \alpha \).

We thus have a formal system with least and greatest fixpoints, that has a semantics inside the topos of trees, but which does not correctly handle least fixpoints. On the other hand, it was shown by [55] that the interpretation of guarded polynomial (i.e. first-order) recursive types in \( S \) induces final coalgebras for the corresponding polynomial functors on the category \( \text{Set} \) of usual sets and functions. This applies e.g. to streams and colists. Hence, it makes sense to think of interpreting least fixpoint formulae over such types externally, in \( \text{Set} \).
The Constant Type Modality. Figure 2 represents adjoint functors $\Gamma : S \to \text{Set}$ and $\Delta : \text{Set} \to S$. To correctly handle least fixpoints $\mu \alpha \varphi : A$, we would like to see them as subsets of $[A]$ in $\text{Set}$ rather than subobjects of $[A]$ in $S$. On the other hand, the internal semantics in $S$ is still necessary to handle definitions by guarded recursion. We navigate between the internal semantics in $S$ and the external semantics in $\text{Set}$ via the adjunction $\Delta \dashv \Gamma$. This adjunction induces a comonad $\Delta \Gamma$ on $S$, which is represented in the guarded $\lambda$-calculus of [20] by the constant type modality $\Box$. This gives coinductive versions of guarded recursive types, e.g. $\text{Str} A := \Box \text{Str}^\varnothing A$ for streams and $\text{CoList} A := \Box \text{CoList}^\varnothing A$ for colists, which allow for productive but not causal programs [20, Ex. 1.10.(3)].

Each formula gets two interpretations: $[\varphi]$ in $S$ and $\{\varphi\}$ in $\text{Set}$. The external semantics $\{\varphi\}$ handles least fixpoints in the standard set-theoretic way, thus the two interpretations differ in general. But we do have $\{\varphi\} = \Gamma [\varphi]$ when $\varphi$ is safe (Def. 6.5), that is, when $\varphi$ describes a safety property. We have a modality $[\text{box}]\varphi$ which lifts $\varphi : A$ to $\Box A$. By defining $\Box [\varphi] := \Delta \{\varphi\}$, we correctly handle the least fixpoints which are guarded by a $[\varphi]$ modality. When $\varphi$ is safe, we can navigate between $\Box A$ and $[\text{box}]\varphi$, and $\Box A \setminus \{\varphi\}$, thus making available the comonad structure of $\Box$ on $[\text{box}]\varphi$. Note that $[\text{box}]$ is unrelated to $\Box$.

Approximating Least Fixpoints. For proving liveness properties on functions defined by guarded recursion, one needs to navigate between e.g. $[\text{box}]\varphi$ and $\Diamond \varphi$, while $\Diamond \varphi$ is in general unsafe. The fixpoint $\Diamond \varphi = \mu \alpha \varphi \lor \Diamond \alpha$ is alternation-free (see e.g. [17, §4.1]). This implies that $\Diamond \varphi$ can be seen as the supremum of the $\Diamond^m \varphi$ for $m \in \mathbb{N}$, where each $\Diamond^m \varphi$ is safe when $\varphi$ is safe. More generally, we can approximate alternation-free $\mu \alpha \varphi$ by their finite unfoldings $\varphi^m(\bot)$, à la Kleene. We extend the logic with finite iterations $\mu^k \alpha \varphi$, where $k$ is an iteration variable, and where $\mu^k \alpha \varphi$ is seen as $\varphi^k(\bot)$. Let $\Diamond^k \varphi := \mu^k \alpha. \varphi \lor \Diamond \alpha$. If $\varphi$ is safe then so is $\Diamond^k \varphi$. For safe $\varphi$, $\psi$, we have the following refinement typings for the guarded recursive $\text{map}^\varnothing$ and its coinductive lift $\text{map}$:

- \text{map}^\varnothing : (\{B \mid \psi\} \to \{A \mid \varphi\}) \to \{\text{Str}^\varnothing B \mid \Diamond^k \text{hd} \psi\} \to \{\text{Str}^\varnothing A \mid \Diamond^k \text{hd} \psi\}
- \text{map} : (\{B \mid \psi\} \to \{A \mid \varphi\}) \to \{\text{Str} B \mid [\text{box}] \Diamond^k \text{hd} \psi\} \to \{\text{Str} A \mid [\text{box}] \Diamond^k \text{hd} \psi\}

3 The Pure Calculus

Our system lies on top of the guarded $\lambda$-calculus of [20]. We briefly review it here. We consider values and terms from the grammar given in Fig. 3 (left). In
both box$\sigma(M)$ and prev$\sigma(E)$, $\sigma$ is a delayed substitution of the form $\sigma = [x_1 \mapsto M_1, \ldots, x_k \mapsto M_k]$ and such that box$\sigma(M)$ and prev$\sigma(E)$ bind $x_1, \ldots, x_k$ in $M$.

We use the following conventions of [20]: box$(M)$ and prev$(M)$ (without indicated substitution) stand resp. for box$_{[]}((M)$ and prev$_{[]}((M)$ i.e. bind no variable of $M$.

Moreover, box$\sigma(M)$ stands for box$[x_1 \mapsto x_1, \ldots, x_k \mapsto x_k](M)$ where $x_1, \ldots, x_k$ is a list of all free variables of $M$, and similarly for prev$(M)$. We consider the weak call-by-name reduction of [20], recalled in Fig. 3 (right).

Pure types (notation $A, B$, etc.) are the closed types over the grammar

$$A ::= \ 1 \mid A + A \mid A \times A \mid A \rightarrow A \mid \text{▸} A \mid X \mid \text{Fix}(X).A \mid [A]$$

where, (1) in the case Fix$(X).A$, each occurrence of $X$ in $A$ must be guarded by a ▸, and (2) in the case of [A], the type $A$ is closed (i.e. has no free type variable).

Guarded recursive types are built with the fixpoint constructor Fix$(X).A$, which allows for $X$ to appear in $A$ both at positive and negative positions, but only under a ▸. In this paper we shall only consider positive types.

Example 3.1. We can code a finite base type $B = \{b_1, \ldots, b_n\}$ as a sum of unit types $\sum_{i=1}^{n} 1 = 1 + (\cdots + 1)$, where the $i$th component of the sum is intended to represent the element $b_i$ of $B$. At the term level, the elements of $B$ are represented as compositions of injections $in_{j_1}(in_{j_2}(\cdots in_{j_i}(\langle \rangle)))$. For instance, Booleans are represented by $\text{Bool} := 1 + 1$, with $tt := in_{0}(\langle \rangle)$ and $ff := in_{1}(\langle \rangle)$.

Example 3.2. Besides streams (Str$^S_A$), colists (CoList$^S_A$), conatural numbers (CoNat$^S_A$) and infinite binary trees (Tree$^S_A$), we consider a type Res$^S_A$ of resumptions (parametrized by I, 0) adapted from [48], and a higher-order recursive type Rou$^S_A$, used in Martin Hofmann’s breadth-first tree traversal (see e.g. [10]):

$$\begin{align*}
\text{Tree}^S_A & := \text{Fix}(X).A \times (\text{▸} X \times \text{▸} X) & \text{CoNat}^S_A & := \text{Fix}(X). \ 1 + \text{▸} X \\
\text{Res}^S_A & := \text{Fix}(X).A + (I \rightarrow (0 \times \text{▸} X)) & \text{Rou}^S_A & := \text{Fix}(X). \ 1 + ((\text{▸} X \rightarrow \text{▸} A) \rightarrow A)
\end{align*}$$

Some typing rules of the pure calculus are given in Fig. 4 where a pure type $A$ is constant if each occurrence of ▸ in $A$ is guarded by a ◦ modality. The omitted rules are the standard ones for simple types with finite sums and products (A).
Example 3.3. Figure 1 defines some operations on guarded streams. On other types of Ex. 3.2, we have e.g., the constructors of colists \( \text{Nil}^S := \text{fold}(\text{in}_0(\langle \rangle)) \) : CoList\(^S\) A and \( \text{Cons}^S := \lambda x.\lambda xs.\text{fold}(\text{in}_1(\langle x, xs \rangle)) : A \rightarrow \text{CoList}^S A \rightarrow \text{CoList}^S A \).

Infinite binary trees Tree\(^S\) A have operations \( \text{son}^S_d : \text{Tree}^S A \rightarrow \text{Tree}^S A \) for \( d \in \{ \ell, r \} \). Node\(^S\) : \text{A} \rightarrow \text{Tree}^S A \rightarrow \text{Tree}^S A \rightarrow \text{Tree}^S A \) and \( \text{label}^S : \text{Tree}^S A \rightarrow A \).

Example 3.4. Coinductive types are guarded recursive types under a \( \square \). For instance Str\(^A\) := \( \square \)Str\(^A\) A, CoList\(^A\) := \( \square \)CoList\(^A\) A, CoNat\(^A\) := \( \square \)CoNat\(^A\) and Res\(^A\) := \( \square \)Res\(^A\) A, with \( A, I, 0 \) constant. Basic operations on guarded types lift to coinductive ones. For instance

\[
\text{Cons} := \lambda x.\lambda xs.\text{box}_x(\text{Cons}^S x \text{next}(\text{unbox} s)) : A \rightarrow \text{Str} A \rightarrow \text{Str} A
\]

\[
\text{hd} := \lambda s.\text{hd}^S (\text{unbox} s) : \text{Str} A \rightarrow A
\]

\[
\text{tl} := \lambda s.\text{box}_x(\text{prev}_v(\text{tl}^S (\text{unbox} s))) : \text{Str} A \rightarrow \text{Str} A
\]

These definitions follow a general pattern to lift a function over a guarded recursive type into one over its coinductive version, by performing an \( \eta \)-expansion with some box and unbox inserted in the right places. For example, one can define the map function on coinductive streams as:

\[
\text{map} := \lambda f.\lambda s.\text{box}_x(\text{map}^S f (\text{unbox} s)) : (B \rightarrow A) \rightarrow \text{Str} B \rightarrow \text{Str} A
\]

4 A Temporal Modal Logic

We present here a logic of (modal) temporal specifications. We focus on syntactic aspects. The semantics is discussed in [7]. For the moment the logic has only one form of fixpoints (\( \mu \alpha \varphi \)). It is extended with least fixpoints (\( \mu \alpha \varphi \)) in [6].

Manysorted Modal Temporal Formulae. The main ingredient of this paper is the logical language we use to annotate pure types when forming refinement types. This language, that we took with minor adaptations from [33], is manysorted: for each pure type \( A \) we have formulae \( \varphi \) of type \( A \) (notation \( \vdash \varphi : A \)). The formulation rules of formulae are given in Fig. 5.

Example 4.1. Given a finite base type \( B = \{ b_1, \ldots, b_n \} \) as in Ex. 4.1, with element \( b_i \) represented by \( \text{in}_{j_1}(\text{in}_{j_2}(\ldots\text{in}_{j_n}(\langle \rangle))) \), the formula \( [\text{in}_{j_1}] [\text{in}_{j_2}] \ldots [\text{in}_{j_n}] T \) represents the singleton subset \( \{ b_i \} \) of \( B \). On \( \text{Bool} \), we have the formulae \( \text{tt} := [\text{in}_0] T \) and \( \text{ff} := [\text{in}_1] T \) representing resp. \( \text{tt} \) and \( \text{ff} \).
(a) The formula \( \text{hd}[a] \Rightarrow \bigcirc \text{hd}[b] \) means that if the head of a stream is \( a \), then its second element (the head of its tail) should be \( b \).

(b) On colists, we let \( \text{hd}[a] := \text{fold}[\text{in}_1][\pi_0][\phi] \) and \( \bigcirc \phi := \text{fold}[\text{in}_1][\pi_1][\text{next}\psi] \).

(c) On (guarded) infinite binary trees over \( A \), we also have a modality \( \text{lb}[\phi] := \text{fold}[\pi_0][\phi] : \text{Tree}^6_A \) (provided \( \phi : A \)).

We just mention here that \( \text{ev}(\text{hd}) \) is contravariant in its first argument. Note that \( \text{ev}(\text{hd}) \) can only be formed for closed \( \phi \).

Example 4.3. (a) The modality \( \Box \) makes it possible to express a range of safety properties. For instance, assuming \( \phi, \psi : \text{Str}^6_A \), the formula \( \Box(\psi \Rightarrow \bigcirc \phi) \) is intended to hold on a stream \( s = (s_i : i \geq 0) \) iff, for all \( n \in \mathbb{N} \), if \( (s_i : i \geq n) \) satisfies \( \psi \), then \( (s_i : i \geq n + 1) \) satisfies \( \phi \).

(b) The modality \( \Box \) has its two \( \text{CTL} \)-like variants on \( \text{Tree}^6_A \), namely \( \forall \Box \phi := \nu \alpha. \phi \land (\bigcirc \phi \land \bigcirc \alpha) \) and \( \exists \Box \phi := \nu \alpha. \phi \land (\bigcirc \phi \lor \bigcirc \alpha) \). Assuming \( \psi : A \), \( \forall \Box \text{lb}[\phi] \psi \) is intended to hold on a tree \( t : \text{Tree}^6_A \) iff all node-labels of \( t \) satisfy \( \psi \), while \( \exists \Box \text{lb}[\phi] \psi \) holds on \( t \) iff \( \psi \) holds on all nodes of some infinite path from the root of \( t \).

**Fig. 5.** Formation Rules of Formulae (where \( A, B \) are pure types).
Modal Theories. Formulae are equipped with a modal deduction system which enters the type system via a subtyping relation (\(\sqsubseteq\)). For each pure type \(A\), we have an intuitionistic theory \(\vdash A\) (the general case) and a classical theory \(\vdash c A\) (which is only assumed under \(\sqsupset\)). Formulated in Fig. 6 and Table 2 (where we also give properties of the derived modalities \([\text{hd}],[\ldotp]\)), in any case, \(\vdash c A\). is only defined when \(\vdash \varphi : A\) (and so when \(\varphi\) has no free propositional variable).

Fixpoints \(\nu \alpha \varphi\) are equipped with their usual Kozen axioms [47]. The atomic modalities \([\pi],[\text{fold}],[\text{next}],[\text{in}],[\text{box}]\) have deterministic branching (see Fig. 12, section 7). We can get the axioms of the intuitionistic (normal) modal logic \(\text{IK}\) [61] (see also e.g. [65,53]) for \([\pi],[\text{fold}],[\text{box}]\) but not for \([\text{in}]\) nor for the intuitionistic \([\text{next}]\). For \([\text{next}]\), in the intuitionistic case this is due to semantic issues with step indexing (discussed in section 7) which are absent from the classical case. As for \([\text{in}]\), we have a logical theory allowing for a coding of finite base types as finite sum types, which allows to derive, for a finite base type \(B\):

\[
\vdash B \quad \bigvee_{a \in B} \left( [a] \wedge \bigwedge_{b \in B} \neg [b] \right)
\]

**Definition 4.4 (Modal Theories).** For each pure type \(A\), the intuitionistic and classical modal theories \(\vdash A \varphi\) and \(\vdash c A \varphi\) (where \(\vdash \varphi : A\)) are defined by mutual induction:

- The theory \(\vdash A\) is deduction for intuitionistic propositional logic augmented with the check-marked (✓) axioms and rules of Table 2 and the axioms and rules of Fig. 3 (for \(\vdash A\)).
- The theory \(\vdash c A\) is \(\vdash A\) augmented with the axioms (P) and (Cₜₙₑₓ) for [next] and with the axiom (CL) (Fig. 6).

For example, we have \(\vdash \text{Str} A \Box \psi \Rightarrow (\psi \wedge \Box \psi)\) and \(\vdash \text{Str} A (\psi \wedge \Box \psi) \Rightarrow \Box \psi\).
Temporal Reflections for Guarded Recursive Types

\[ \vdash^A \phi \Rightarrow \psi \quad \vdash \varphi : A \quad \vdash^A \left[ (\text{ev}(\phi))\varphi \Rightarrow (\text{ev}(\psi))\varphi \right] \quad \vdash^A \left[ (\text{ev}(\psi_0))\varphi \land (\text{ev}(\psi_1))\varphi \Rightarrow (\text{ev}(\psi_0 \lor \psi_1))\varphi \right] \]

\[ \vdash^A ((\varphi \Rightarrow \psi) \Rightarrow \varphi) \Rightarrow \varphi \quad (\text{CL}) \quad \vdash^A \left[ \boxdot \varphi \right] \varphi \quad \vdash^A \left[ \bot \equiv \top \right] \]

\[ \vdash^A \left[ (\text{in}_0)\top \lor (\text{in}_1)\top \land \neg ((\text{in}_0)\top \land (\text{in}_1)\top) \right] \]

\[ \vdash^A \left[ (\text{in}_0)\top \Rightarrow (\neg (\text{in}_0)\varphi \leftrightarrow (\text{in}_0)\neg \varphi) \right] \quad \vdash^A \left[ \nu \alpha \varphi \Rightarrow \varphi[\nu \alpha \varphi/\alpha] \right] \quad \vdash^A \psi \Rightarrow \varphi[\psi/\alpha] \]

Fig. 6. Modal Axioms and Rules.

\[ T \leq |T| \quad A \leq \{A \mid \top\} \quad \{A \mid \varphi\} \leq \{A \mid \psi\} \quad \{\boxdot A \mid \text{box}\varphi\} \leq \{\boxdot A \mid \text{box}\psi\} \]

\[ \{\text{next}\varphi\} \equiv \{\text{next}\varphi\} \quad \{B \rightarrow A \mid \text{ev}(\psi))\varphi\} \equiv \{B \mid \psi\} \rightarrow \{A \mid \varphi\} \]

Fig. 7. Subtyping Rules (excerpt).

5 A Temporally Refined Type System

Temporal refinement types (or types), notation \( T, U, V, \) etc., are defined by:

\[ T, U, V, \ldots := A \mid \{A \mid \varphi\} \mid T + T \mid T \times T \mid T \rightarrow T \mid \text{next} T \]

where \( \vdash \varphi : A \) and, in the case of \( \boxdot T \), the type \( T \) has no free type variable. So types are built from (closed) pure types \( A \) and temporal refinements \( \{A \mid \varphi\} \). They allow for all the type constructors of pure types.

As a refinement type \( \{A \mid \varphi\} \) intuitively represents a subset of the inhabitants of \( A \), it is natural to equip our system with a notion of subtyping. In addition to the usual rules for product, arrow and sum types, our subtyping relation is made of two more ingredients. The first follows the principle that our refinement type system is meant to prove properties of programs, and not to type more programs, so that (say) a type of the form \( \{A \mid \varphi\} \rightarrow \{B \mid \psi\} \) is a subtype of \( A \rightarrow B \). We formalize this with the notion of underlying pure type \( |T| \) of a type \( T \). The second ingredient is the modal theory \( \vdash^A \varphi \) of [4]. The subtyping rules concerning refinements are given in Fig. 4, where \( T \equiv U \) enforces both \( T \leq U \) and \( U \leq T \). The full set of rules is given in Fig. 7 in [4]. Notice that subtyping does not incorporate (un)foiling of guarded recursive types.

Typing for refinement types is given by the rules of Fig. 3 together with the rules of extended to refinement types, where \( T \) is constant if \( |T| \) is constant. Modalities \( \pi_i, \text{in}_i, \text{fold} \) and \( \text{ev}(-) \) (but not \( \text{next} \)) have introduction rules extending those of the corresponding term formers.
Example 5.2. Get the first derived rule below, from which we can deduce the second one:

Example 5.1. Since \( g \) is a function which takes two guarded streams and interleaves them:

Fig. 8. Typing Rules for Refined Modal Types.

Example 5.1. Since \( \varphi \Rightarrow \psi \Rightarrow (\varphi \land \psi) \) and using two times the rule (MP), we get the first derived rule below, from which we can deduce the second one:

Example 5.2. We have the following derived rules:

Example 5.3. We have \( \text{Cons}^\delta : A \rightarrow \{\text{Str}^\delta A \mid \varphi\} \rightarrow \{\text{Str}^\delta A \mid \lor \varphi\} \) as well as \( \text{tl}^\delta : \{\text{Str}^\delta A \mid \varphi\} \rightarrow \{\text{Str}^\delta A \mid \lor \varphi\} \).

Example 5.4. “Always” (\( \Box \)) on Guarded Streams. The refined types of \( \text{Cons}^\delta \), \( \text{hd}^\delta \), \( \text{tl}^\delta \) and \( \text{map}^\delta \) mentioned in \( 2 \) are easy to derive. We also have the type

for the \( \text{merge}^\delta \) function which takes two guarded streams and interleaves them:

merge^\delta : \text{Str}^\delta A \rightarrow \text{Str}^\delta A \rightarrow \text{Str}^\delta A
:= \text{fix}(g).\lambda s_0.\lambda s_1. (\text{hd}^\delta s_0) \circ \text{next}((\text{hd}^\delta s_1) \circ g (\text{tl}^\delta s_0) \circ (\text{tl}^\delta s_1)))

6 The Full System

The system presented so far has only one form of fixpoints in formulae (\( \nu \alpha \varphi \)). We now present our full system, which also handles least fixpoints (\( \mu \alpha \varphi \)) and thus liveness properties. A key role is played by polynomial guarded recursive types, that we discuss first.
Strictly Positive and Polynomial Types. Strictly positive types (notation $P^+$, $Q^+$, etc.) are given by

$$P^+ ::= A \mid X \mid ▶P^+ \mid P^+ + P^+ \mid P^+ × P^+ \mid \text{Fix}(X).P^+ \mid B → P^+$$

where $A$, $B$ are (closed) constant pure types. Strictly positive types are a convenient generalization of polynomial types. A guarded recursive type Fix $(X).P(X)$ is polynomial if $P(X)$ is induced by

$$P(X) ::= A \mid ▶X \mid P(X) + P(X) \mid P(X) × P(X) \mid B → P(X)$$

where $A$, $B$ are (closed) constant pure types. Note that if Fix $(X).P(X)$ is polynomial, $X$ cannot occur on the left of an arrow ($→$) in $P(X)$. We say that Fix $(X).P(X)$ (resp. $P^+$) is finitary polynomial (resp. finitary strictly positive) if $B$ is a finite base type (see Ex. 3.1) in the above grammars. The set-theoretical counterpart of our polynomial recursive types are the exponent polynomial functors of [34], which all have final Set-coalgebras (see e.g. [34] Cor. 4.6.3).

Example 6.1. For $A$ a constant pure type, e.g. $\text{Str}^8 A$, $\text{CoList}^8 A$ and $\text{Tree}^8 A$ as well as $\text{Str}^8(\text{Str} A)$, $\text{CoList}^8(\text{Str} A)$ and $\text{Res}^8 A$ (with 1, 0 constant) are polynomial. More generally, polynomial types include all recursive types Fix $(X).P(X)$ where $P(X)$ is of the form $\sum_{i=0}^n A_i × (▶X)^{B_i}$ with $A_i$, $B_i$ constant. The non-strictly positive recursive type $\text{Rou}^8 A$ of Ex. 3.2 used in Hofmann’s breadth-first traversal (see e.g. [10]), is not polynomial.

The Full Temporal Modal Logic. We assume given a first-order signature of iteration terms (notation $t$, $u$, etc.), with iteration variables $k$, $ℓ$, etc., and for each iteration term $t(k_1, \ldots, k_m)$ with variables as shown, a given primitive recursive function $[t_k] : \mathbb{N}^m → \mathbb{N}$. We assume a term $0$ for $0 ∈ \mathbb{N}$ and a term $k+1$ for the successor function $n ∈ \mathbb{N} → n + 1 ∈ \mathbb{N}$.
The formulae of the full temporal modal logic extend those of Fig. 5 with least fixpoints $\mu \alpha \varphi$ and with approximated fixpoints $\mu^t \alpha \varphi$ and $\nu^t \alpha \varphi$ where $t$ is an iteration term. The additional formation rule for formulae are given in Fig. 9. We use $\theta$ as a generic notation for $\mu$ and $\nu$. Least fixpoints $\mu \alpha \varphi$ are equipped with their usual Kozen axioms. In addition, iteration formulae $\nu^t \alpha \varphi(\alpha)$ and $\mu^t \alpha \varphi(\alpha)$ have axioms expressing that they are indeed iterations of $\varphi(\alpha)$ from resp. $\top$ and $\bot$. A fixpoint logic with iteration variables was already considered in [68].

Definition 6.2 (Full Modal Theories). The full intuitionistic and classical modal theories (still denoted $\vdash$ and $\vdash^c$) are defined by extending Def. 4.4 with the axioms and rules of Fig. 10.

Example 6.3. Least fixpoints allow us to define liveness properties. On streams and colists, we have

$$3 \varphi := \mu \alpha. \varphi \lor \boxdot \alpha$$

and

$$\varphi U \psi := \mu \alpha. \psi \lor (\varphi \land \boxdot \alpha).$$

On trees, we have the CTL-like

$$\exists 3 \varphi := \mu \alpha. \varphi \lor (\boxdot \ell \alpha \lor \boxdot r \alpha)$$

and

$$\forall 3 \varphi := \mu \alpha. \varphi \lor (\boxdot \ell \alpha \land \boxdot r \alpha).$$

The formula $\exists 3 \varphi$ is intended to hold on a tree if there is a finite path which leads to a subtree satisfying $\varphi$, while $\forall 3 \varphi$ is intended to hold if every infinite path crosses a subtree satisfying $\varphi$.

Remark 6.4. On finitary trees (as in Ex. 6.1 but with $A_i$, $B_i$ finite base types), we have all formulae of the modal $\mu$-calculus. For this fragment, satisfiability is decidable (see e.g. [17]), as well as the classical theory $\vdash^c$ by completeness of Kozen’s axiomatization [74] (see [63] for completeness results on fragments of the $\mu$-calculus).

The Safe and Smooth Fragments. We now discuss two related but distinct fragments of the temporal modal logic. Both fragments directly impact the refinement type system by allowing for more typing rules.

The safe fragment plays a crucial role, because it reconciles the internal and external semantics of our system (see §7). It gives subtyping rules for $\blacksquare$ (Fig. 11), which makes available the comonad structure of $\blacksquare$ on $\boxdot \varphi$ when $\varphi$ is safe.

Definition 6.5 (Safe Formula). Say $\alpha_1 : A_1, \ldots, \alpha_n : A_n \vdash \varphi : A$ is safe if

(i) the types $A_1, \ldots, A_n, A$ are strictly positive, and

(ii) for each occurrence in $\varphi$ of a modality $[\text{ev(}\psi\text{)\text{]}\text{]}$, the formula $\psi$ is closed, and

(iii) each occurrence in $\varphi$ of a least fixpoint ($\mu \alpha (\neg \alpha)$) and of an implication ($\alpha \Rightarrow \beta$) is guarded by a $\blacksquare$.

Note that the safe restriction imposes no condition on approximated fixpoints $\theta^t \alpha$. Recalling that the theory under a $[\text{box}]$ is $\vdash^c$, the only propositional connectives accessible to $\vdash^c$ in safe formulae are those on which $\vdash^c$ and $\vdash^A$ coincide. The formula $[\neg \text{nil}] = [\text{fold}][\text{in}]\top$ is safe. Moreover:

Example 6.6. Any formula without fixpoint nor $[\text{ev(}\neg \text{)\text{]}\text{]}$ is equivalent in $\vdash^c$ to a safe one. It $\varphi$ is safe, then so are $[\text{hd}]\varphi$, $[\text{lbl}]\varphi$, as well as $\Delta \varphi$ (for $\Delta \in \{\boxdot, \forall \alpha, \exists \alpha\}$) and $[\text{box}]\Delta \varphi$ (for $\Delta \in \{\boxdot, \exists \alpha, \forall \alpha\}$).
Definition 6.7 (Smooth Formula). A formula $\alpha_1 : A_1, \ldots, \alpha_n : A_n \vdash \varphi : A$ is smooth if

(i) the types $A_1, \ldots, A_n, A$ are finitary strictly positive, and
(ii) for each occurrence in $\varphi$ of a modality $[\text{ev}(\psi)]$, the formula $\psi$ is closed, and
(iii) $\varphi$ is alternation-free: for $\theta, \theta' \in \{\mu, \nu\}$, (1) if $\theta \beta_0 \psi_0$ is a subformula of $\varphi$, and $\theta' \beta_1 \psi_1$ is a subformula of $\psi_0$ s.t. $\beta_0$ occurs free in $\psi_1$, then $\theta = \theta'$, (2) if some $\alpha_i$ occurs in two subformulas $\theta_0 \psi_0$ and $\theta_1 \psi_1$ of $\varphi$, then $\theta = \theta'$, and (3) if some $\alpha_i$ occurs in a subformula $\theta' \beta \psi$ of $\varphi$, then $\alpha_i$ occurs $\beta$ times.

Our notion of alternation freedom is adapted from [17], in which propositional (fixpoint) variables are always positive. Note that the smooth restriction imposes no further conditions on approximated fixpoints $\theta^\alpha$. In the smooth fragment, greatest and least fixpoints can be thought about resp. as $\bigwedge_{m \in \mathbb{N}} \varphi^m(\top)$ and $\bigvee_{m \in \mathbb{N}} \varphi^m(\bot)$.

Iteration terms allow for formal reasoning about such unfoldings. Assuming $[\varepsilon] = m \in \mathbb{N}$, the formula $\nu^\varepsilon \alpha \varphi(\alpha)$ (resp. $\mu^\varepsilon \alpha \varphi(\alpha)$) can be read as $\varphi^m(\top)$ (resp. $\varphi^m(\bot)$). This gives the rules ($\nu$-I) and ($\mu$-E) (Fig. 11), which allow for reductions to the safe case (see examples in [8]).

Remark 6.8. It is well-known (see e.g. [17, §4.1]) that on finitary trees (see Rem. 6.4) the alternation-free fragment is equivalent to Weak MSO (MSO with second-order variables restricted to finite sets). In the case of streams $\text{Str B}$ (for a finite base type B), Weak MSO is in turn equivalent to the full modal $\mu$-calculus. In particular, the alternation-free fragment contains all the flat fixpoints of [63] and thus LTL on $\text{Str B}$ and CTL on $\text{Tree B}$ and on $\text{Res B}$ with I, O, B finite base types. A typical property on $\text{Tree B}$ which cannot be expressed with alternation-free formulae is “there is an infinite path with infinitely many occurrences of b” for a fixed $b : B$ (see e.g. [17, §2.2]).

Example 6.9. Any formula without fixpoint nor $[\text{ev}(\cdot)]$ is smooth. It $\varphi$ is smooth, then so are $[\text{hd}] \varphi$, $[\text{lb}] \varphi$ and $\bigtriangleup \varphi$ for $\bigtriangleup \in \{\square, \exists \square, \exists \Diamond, \Diamond \Diamond, \forall \Diamond\}$.

The Full System. We extend the types of [5] with universal quantification over iteration variables ($\forall k \cdot T$). The type system of [5] is extended with the rules of Fig. 11.

Example 6.10. The logical rules of Fig. 10 give the following derived typing rules (where $\beta$ Pos $\gamma$):

\[
\begin{align*}
(\mu\text{-}I) & \quad \mathcal{E} \vdash M : \{\llbracket A \mid [\text{box}]\gamma[\mu^\varepsilon \alpha \varphi(\beta)]\}\} \\
\frac{\mathcal{E} \vdash M : \{\llbracket A \mid [\text{box}]\gamma[\mu \alpha \varphi(\beta)]\}\}}{\mathcal{E} \vdash M : \{\llbracket A \mid [\text{box}]\gamma[\nu \alpha \varphi(\beta)]\}\}} \\
(\nu\text{-}E) & \quad \mathcal{E} \vdash M : \{\llbracket A \mid [\text{box}]\gamma[\nu^\varepsilon \alpha \varphi(\beta)]\}\} \\
\end{align*}
\]
∀k • T \equiv \forall k \cdot T

(\forall I) \quad \mathcal{E} \vdash M : T \quad \mathcal{E} \vdash M : \forall k \cdot T

(\forall CI) \quad \mathcal{E} \vdash M : T[0/k] \quad \mathcal{E} \vdash M : T[k+1/k]

(\forall E) \quad \mathcal{E} \vdash M : \forall k \cdot T

\mathcal{E} \vdash M : T[\ell/k]

\mathcal{E} \vdash M : \forall k \cdot T

\mathcal{E} \vdash M : T[\ell/k]

(\mu E) \quad \mathcal{E} \vdash M : \{A | [\mathbf{box}] [\gamma [\mu [\alpha \psi / \beta]]]\}

\mathcal{E}, x : \{A | [\mathbf{box}] [\gamma [\alpha \psi / \beta]]\} \vdash N : U

\mathcal{E} \vdash N[M/x] : U

\begin{align*}
\text{safe} & \quad \{A | [\mathbf{box}] [\varphi]\} \equiv \{A | \varphi\} \\
\forall k \cdot T & \equiv \forall k \cdot T
\end{align*}

\textbf{Fig. 11.} Extended (Sub)Typing Rules for Refinement Types (where } k \text{ is not free in } \mathcal{E} \text{ in } (\forall I) \& (\forall CI), \ell \text{ is fresh in } (\forall I) \& (\mu E), \theta \alpha \psi \text{ and } \gamma \text{ are smooth, and } \beta \text{ Pos } \gamma.

\section{Semantics}

We present the main ingredients of the semantics of our type system. We take as base the denotational semantics of guarded recursion in the topos of trees.

\textbf{Denotational Semantics in the Topos of Trees.} The topos of trees } \mathcal{S} \text{ provides a natural model of guarded recursion [13]. Formally, } \mathcal{S} \text{ is the category of presheaves over } (\mathbb{N} \setminus \{0\}, \leq). \text{ In words, the objects of } \mathcal{S} \text{ are indexed sets } X = (X(n))_{n>0} \text{ equipped with restriction maps } r_n^X : X(n+1) \to X(n). \text{ Excluding 0 from the indexes is a customary notational convenience ([13]). The morphisms from } X \text{ to } Y \text{ are families of functions } f = (f_n : X(n) \to Y(n))_{n>0} \text{ which commute with restriction, that is } f_n \circ r_n^X = r_Y^X \circ f_{n+1}. \text{ As any presheaf category, } \mathcal{S} \text{ has (pointwise) limits and colimits, and is Cartesian closed (see e.g. [52, §1.6]). We write } \Gamma : \mathcal{S} \to \text{Set for the global section functor, which takes } X \text{ to } \mathcal{S}[1, X], \text{ the set of morphisms } 1 \to X \text{ in } \mathcal{S}, \text{ where } 1 = (\{\bullet\})_{n>0} \text{ is terminal in } \mathcal{S}.

A typed term } \mathcal{E} \vdash M : T \text{ is to be interpreted in } \mathcal{S} \text{ as a morphism}

\[ [M] : [[\mathcal{E}]] \to [[T]] \]

where } [[\mathcal{E}]] = [[T_1]] \times \cdots \times [[T_n]] \text{ for } \mathcal{E} = x_1 : T_1, \ldots, x_n : T_n. \text{ In particular, a closed term } M : T \text{ is to be interpreted as a global section } [M] \in \Gamma[[T]]. \text{ The } x / + / \to \text{ fragment of the calculus is interpreted by the corresponding structure in } \mathcal{S}. \text{ The } \blacktriangle \text{ modality is interpreted by the functor } \blacktriangle : \mathcal{S} \to \mathcal{S} \text{ of [13]. This functor shifts indexes by 1 and inserts a singleton set } 1 \text{ at index 1. The term constructor } \text{next} \text{ is interpreted by the natural map with component } \text{next}^X : X \to \blacktriangle X \text{ as in}

\begin{align*}
\text{next}^X & \quad X \quad X_1 \xleftarrow{r_1^X} X_2 \leftarrow \cdots \leftarrow X_n \xleftarrow{r_n^X} X_{n+1} \leftarrow \cdots \\
\blacktriangle X & \quad 1 \quad 1 \xleftarrow{r_1^X} X_1 \leftarrow \cdots \leftarrow X_{n-1} \xleftarrow{r_n^{n-1}} X_n \leftarrow \cdots
\end{align*}
The guarded fixpoint combinator \textit{fix} is interpreted by the morphism $\textit{fix}^X : X \times X \rightarrow X$ of [13] Thm. 2.4.

The constant type modality $\Box$ is interpreted as the comonad $\Delta \Gamma : \text{Set} \rightarrow \text{Set}$, where the left adjoint $\Delta : \text{Set} \rightarrow \text{S}$ is the constant object functor, which takes a set $S$ to the constant family $(S)_{n>0}$. In words, all components $[[\Box A]](n)$ are equal to $\Gamma[A]$, and the restriction maps of $[[\Box A]]$ are identities. In particular, a global section $x \in \Gamma[[\Box A]]$ is a constant family $(x_n)_{n}$ describing a unique global section $x_{n+1}(\bullet) = x_n(\bullet) \in \Gamma[A]$. We refer to [20] and [12] for the interpretation of $\text{prev}$, $\text{box}$ and $\text{unbox}$. Just note that the unit $\eta : \text{Id}_{\text{Set}} \rightarrow \Gamma \Delta$ is an iso.


**External Semantics.** Mogelberg [55] has shown that for polynomial types such as $\text{Str}^B$ with $B$ a constant type, the set of global sections $\Gamma[[\text{Str}^B]]$ is equipped with the usual final coalgebra structure of streams over $B$ in Set. To each polynomial recursive type $\text{Fix}(X).P(X)$, we associate a polynomial functor $P_{\text{Set}} : \text{Set} \rightarrow \text{Set}$ in the obvious way.

**Theorem 7.1 ([55] (see also [20])).** If $\text{Fix}(X).P(X)$ is polynomial, then the set $\Gamma[[\text{Fix}(X).P(X)]]$ carries a final $\text{Set}$-coalgebra structure for $P_{\text{Set}}$.

We devise a $\text{Set}$ interpretation $[[\varphi]] \in \mathcal{P}(\Gamma[A])$ of formulae $\varphi : A$. We rely on the (complete) Boolean algebra structure of powersets for propositional connectives and on Knaster-Tarski Fixpoint Theorem for fixpoints $\mu$ and $\nu$. The interpretations of $\nu^\alpha \varphi(\alpha)$ and $\mu^\alpha \varphi(\alpha)$ (for $\alpha$ closed) are defined to be the interpretations resp. of $\varphi^\alpha(\top)$ and $\varphi^\alpha(\bot)$, where e.g. $\varphi^\alpha(\top) \equiv \top$ and $\varphi^\alpha(\bot) \equiv \varphi(\varphi^\alpha(\bot))$. We give the cases of the atomic modalities in Fig. 12 (where for simplicity we assume formulae to be closed). It can be checked that, when restricting to polynomial types, one gets the coalgebraic semantics of [53] (with sums as in [34]) extended to fixpoints.

**Internal Semantics of Formulae.** We would like to have adequacy w.r.t. the external semantics of formulae, namely that given $M : \{ A \mid \varphi \}$, the global section $[[M]] \in \Gamma[[A]]$ satisfies $[[\varphi]] \in \mathcal{P}(\Gamma[[A]])$ in the sense that $[[M]] \in [[\varphi]]$. But in general we can only have adequacy w.r.t. an internal semantics $[[\varphi]] \in \text{Sub}([A])$.
of formulae \( \varphi : A \). We sketch it here. First, \( \text{Sub}(X) \) is the (complete) Heyting algebra of subobjects of an object \( X \) of \( \mathcal{S} \). Explicitly, we have \( \mathcal{S} = (S(n))_{n \in \mathbb{N}} \subseteq \text{Sub}(X) \) if for all \( n > 0 \), \( S(n) \subseteq X(n) \) and \( r^X_\alpha(t) \in S(n) \) whenever \( t \in S(n + 1) \).

For propositional connectives and fixpoints, the internal \([-]\) is defined similarly as the external \([[-]]\), but using (complete) Heyting algebras of subobjects rather than (complete) Boolean algebras of subsets.

As for modalities, let \( [\Delta] \) be of the form \([\pi_i], [\text{in}_i], [\text{next}] \) or \([\text{fold}]\), and assume \([\Delta]\varphi : B \) whenever \( \varphi : A \). Standard topos theoretic constructions give posets morphisms \([\Delta] : \text{Sub}(A) \to \text{Sub}(B)\) such that \([\pi_i]\), \([\text{fold}]\) are maps of Heyting algebras, \([\text{in}_i]\) preserves \( \lor, \bot \) and \( \land \), while \([\text{next}]\) preserves \( \land, \top \) and \( \lor \). With \([\Delta]\varphi := \{\|\Delta\|([\varphi])\}, \) all the axioms and rules of Table 2 are validated for these modalities. To handle guarded recursion, it is crucial to have \([\text{next}]\varphi := \text{true} \) at time 1, independently from \( \varphi \). As a consequence, \( [\text{next}] \) and \( \bigcirc \) do not validate axiom (P) (Table 2), and \( \bigcirc [\text{hd}]\varphi \) can “lie” about the next time step. We let \([\text{box}]\varphi := \Delta([\varphi])\).

The modality \( \text{ev}(\psi) \) is a bit more complex. For \( \psi : B \) and \( \varphi : A \), the formula \( \text{ev}(\psi)\varphi \) is interpreted as a logical predicate in the sense of \([32],[9.2 \& \text{Prop. 9.2.4}]\). The idea is that for a term \( M : \{B \to A \mid \text{ev}(\psi)\varphi\} \), the global section \( \text{ev} \circ (\{M\}, x) \in \Gamma[A] \) should satisfy \( \varphi \) whenever \( x \in \Gamma[B] \) satisfies \( \psi \). We refer to \([32]\) for details.

Our semantics are both correct w.r.t. the full modal theories of Def. 6.2.

**Lemma 7.2.** If \( \vdash^\alpha \varphi \) then \( \|\varphi\| = \{\top\} \). If \( \vdash^A \varphi \) then \( \|\varphi\| = \{\top\}. \)

**The Safe Fragment.** For \( \alpha \) (positive and) guarded in \( \varphi \), the internal semantics of \( \alpha \varphi \) is somewhat meaningless because \( \mathcal{S} \) has unique guarded fixedpoints \([13],[\S 2.5]\). In particular, the typing \( \alpha \) a s : \( \{\Str^\alpha A \mid \varphi\} \) for arbitrary \( a : A \) and \( \varphi : \Str^\alpha A \) (extending \([2]\)) is indeed verified by the \( \mathcal{S} \) semantics \([-\cdot]-\).

This prevents us from adequacy w.r.t. the external semantics in general. But this is possible for safe formulae since in this case we have:

**Proposition 7.3.** If \( \varphi : A \) is safe then \( \|\varphi\| = \{\top\} \).

Proposition 7.3 gives the subtyping rule \( \{\Box A \mid \text{box}\varphi\} \equiv \{ A \mid \varphi\} \) (Fig. 11), which makes available the coinductive structure of \( \Box \) on \( \text{box}\varphi \) when \( \varphi \) is safe. Recall that in safe formulae, implications can only occur under a \( \Box \) modality and thus in closed subformulae. It is crucial for Prop. 7.3 that infs and sups are pointwise in the subobject lattices of \( \mathcal{S} \), so that conjunctions and disjunctions are interpreted as with the usual classical Kripke semantics (see e.g. \([52],[\S V1.7]\)). This does not hold for implications!

The second key to Prop. 7.3 is the following. For \( L \) a complete lattice, a Scott cocontinuous function \( L \to L \) is a Scott continuous function \( L^{\text{op}} \to L^{\text{op}} \), i.e. which preserves codirected infs. For a safe \( \alpha : A \vdash \varphi : A \), the poset maps \( \{\varphi\} : \text{Sub}(A) \to \text{Sub}(A) \) and \( \{\varphi\} : \mathcal{P}(A) \to \mathcal{P}(A) \) are Scott cocontinuous. The greatest fixpoint \( \nu \alpha \varphi(\alpha) \) can thus be interpreted, both in \( \text{Set} \) and \( \mathcal{S} \), using Kleene’s Fixpoint Theorem, as the infs of the interpretations of \( \varphi^m(\top) \) for \( m \in \mathbb{N} \). This leads to the expected coincidence of the two semantics for safe formulae.
Theorem 7.7 (Adequacy). If \( J \mapsto^{→} xs \) \( s \) pattern matching, e.g. assuming \( \alpha \) and write \( \{ | \} \) with the following conventions. We use the infix notation \( \Gamma \). We exemplified basic manipulations of our system over \( \Gamma \). We have \( \{ | \} \) given a closed smooth \( \nu \). This implies the correctness of the typing rules \((\nu\text{-I})\) and \((\mu\text{-E})\) of Fig. 11.

Lemma 7.4. Given a closed smooth \( \nu\alpha\varphi(\alpha) : A \) (resp. \( \mu\alpha\varphi(\alpha) : A \)), the function \( \{ | \varphi \} \) : \( \mathcal{P}(\Gamma[A]) \to \mathcal{P}(\Gamma[A]) \) is Scott-cocontinuous (resp. Scott-continuous). We have \( \{ | \nu\alpha\varphi(\alpha) \} = \bigcap_{m \in \mathbb{N}} \{ | \varphi^m(T) \} \) (resp. \( \{ | \mu\alpha\varphi(\alpha) \} = \bigcup_{m \in \mathbb{N}} \{ | \varphi^m(\bot) \} \)).

The Realizability Semantics. The correctness of the type system w.r.t. its semantics in \( S \) is proved with a realizability relation.

Definition 7.5 (Realizability). Given a type \( T \) without free iteration variable, a global section \( x \in \Gamma[T] \) and \( n > 0 \), we define the realizability relation \( x \models_n T \) by induction on lexicographically ordered pairs \( (n,T) \) in Fig. 13.

Lemma 7.6. Given types \( T,U \) without free iteration variable, if \( x \models_n U \) and \( U \leq T \) then \( x \models_n T \).

Theorem 7.7 (Adequacy). If \( \triangleright M : T \) where \( T \) has no free iteration variable, then \( [M] \models_n T \) for all \( n > 0 \).

By Thm. 7.7 a program \( M : B \to A \) induces a set-theoretic function \( \Gamma[M] : \Gamma[B] \to \Gamma[A]\circ x \). When \( B \) and \( A \) are polynomial (e.g. streams \( \text{Str}^B \), \( \text{Str}^A \) with \( B, A \) constant), Mogelberg’s Thm. 7.1 says that \( \Gamma[M] \) is a function on the usual final coalgebra for \( B, A \) in \( \text{Set} \) (e.g. the set of usual streams over \( B \) and \( A \)). Moreover, if e.g. \( M : \{ \text{Str} B \mid [\text{box}\psi] \to \{ \text{Str} A \mid [\text{box}\varphi] \} \), then (modulo \( \Gamma \Delta \simeq \text{Id}_\text{Set} \)) given a stream \( x \) that satisfies \( \psi \) (i.e. \( x \in [\psi] \)) the stream \( \Gamma[M](x) \) satisfies \( \varphi \) (i.e. \( \Gamma[M](x) \in [\varphi] \)). See [8] for examples.

8 Examples

We exemplified basic manipulations of our system over [8][8]. We give further examples here. The functions used in our main examples are gathered in Table 3 with the following conventions. We use the infix notation \( a :^\ast s \) for \( \text{Cons}^s a \) \( s \) and write \( [\ast] \) for the empty colist \( \text{Nil}^s \). Moreover, we use some syntactic sugar for pattern matching, e.g. assuming \( s : \text{CoList}^s A \) we write \( \text{case } s \) of \( [\ast] \to N[x :^\ast xs \to M] \) for \( \text{case}(\text{unfold } s) \) of \( (y.N[\cdot]/y)[y.M[\pi_0(y)/x, \pi_1(y)/xs]] \). Most of the
The argument is finite while the second one has an element which satisfies $\varphi$. In addition, if the first argument of $\Gamma$ is finite, then the result has an element which satisfies $\varphi$. The same holds if the first argument is finite while the second one has an element which satisfies $\varphi$.

**Example 8.1 (The Append Function on CoLists).** Our system can derive that $\Gamma[\text{append}]$ returns a non-empty colist if one of its arguments is non-empty. Using $\diamond\text{nil}$ (which says that a colist is finite), we can derive that $\Gamma[\text{append}]$ returns a finite colist if its arguments are both finite. This involves the intermediate typing

$$\forall k. \forall \ell. \emptyset \mapsto \emptyset$$

In addition, if the first argument of $\Gamma[\text{append}]$ has an element which satisfies $\varphi$, then the result has an element which satisfies $\varphi$. The same holds if the first argument is finite while the second one has an element which satisfies $\varphi$. □
Example 8.3 (The Diagonal Function). Consider a stream of streams \( \text{diag} \). The expected refinement types for composition, so that assuming \( \text{tl} \), we have

\[
\text{diag} : (\exists \text{A} | \text{A} \vdash \varphi) \rightarrow (\exists \text{B} | \text{B} \vdash \varphi) \rightarrow (\exists \text{A} | \text{A} \vdash \varphi) 
\]

of \( \text{diag} \) is then the stream \( (\text{diag} \leftarrow \text{tl}) \) is the stream of streams \( (\text{diag} \leftarrow \text{tl}) \). Taking its head thus gives \( \varphi \). Note that \( \text{diag} \) is either \( \text{tl} \) or \( \varnothing \) given \( \varphi \).

Example 8.2 (The Map Function on Streams). The composite modalities \( \Box \triangledown \) and \( \Box \\Diamond \) over streams are read resp. as “infinitely often” and “eventually always”. Provided with a function \( f : \Gamma[B] \rightarrow \Gamma[A] \) taking \( b \in \Gamma[B] \) satisfying \( \psi \) to \( f(b) \in \Gamma[B] \) satisfying \( \varphi \), the function \( \Gamma[\text{map}] \) on set-theoretic streams returns a stream which infinitely often resp. eventually always satisfies \( \varphi \) whenever its stream argument infinitely often resp. eventually always satisfies \( \psi \).

Example 8.3 (The Diagonal Function). Consider a stream of streams \( s \). We have \( s = (s_i | i \geq 0) \) where each \( s_i \) is itself a stream \( s_i = (s_{i,j} | j \geq 0) \). The diagonal of \( s \) is then the stream \( (s_{i,j} | i \geq 0) \). Note that \( s_{i,j} = \text{hd}(\text{tl}(\text{tl}(s_{i,j}))) \). Indeed, \( \text{tl}(s) \) is the stream of streams \( s_k | k \geq i \) so that \( \text{hd}(\text{tl}(s)) \) is the stream \( s_i \) and \( \text{tl}(\text{hd}(\text{tl}(s))) \) is the stream \( s_{i,k} | k \geq i \). Taking its head thus gives \( s_{i,j} \). In the \text{diag} function of Table 3, the auxiliary higher-order function \text{diagaux} \( \hat{=} \) iterates the coinductive \( \text{tl} \) over the head of the stream of streams \( s \). We write \( \circ \) for function composition, so that assuming \( s : \text{Str}^b(\text{Str} A) \) and \( t : \text{Str} A \rightarrow \text{Str} A \), we have (on the coinductive type \( \text{Str} A \))

\[
(\text{hd} \circ t) : \text{Str} A \rightarrow A \quad (\text{hd} \circ t)(\text{hd} \circ t) : A \quad (t \circ \text{tl}) : \text{Str} A \rightarrow \text{Str} A
\]

The expected refinement types for \( \text{diag} \) (Table 4) say that if its argument is a stream whose component streams all satisfy \( \Box \psi \), then \( \Gamma[\text{diag}] \) returns a stream...
whose elements all satisfy \( \varphi \). Also, if the argument of \( \Gamma[\text{diag}] \) is a stream such that eventually all its component streams satisfy \( \Box \varphi \), then it returns a stream which eventually always satisfies \( \varphi \). See \([E.4]\) for details.

**Example 8.4 (A Fair Stream of Booleans).** The non-regular stream \((\text{fb} \ 0 \ 1)\), adapted from \([18,8]\), is of the form \( \text{ff} \cdot \text{tt} \cdot \text{ff} \cdot \text{tt}^2 \cdot \text{ff} \cdot \text{tt}^m \cdot \text{ff} \cdot \text{tt}^{m+1} \cdot \text{ff} \cdots \). It thus contains infinitely many \( \text{tt} \)'s and infinitely many \( \text{ff} \)'s. We indeed have (see \([E.5]\) for details) \((\text{fb} \ 0 \ 1) : \{\text{Str} \ \text{Bool} \mid \boxed{\Box \Box \hdots [\text{tt}]} \land \boxed{\Box \Box \hdots [\text{ff}]}\} \). \( \Box \)

**Example 8.5 (Resumptions).** The type of resumptions \( \text{Res}^6\ A \) (see Ex. \(3.2\)) is adapted from \([18]\). Its guarded constructors are

\[
\begin{align*}
\text{Ret}^6 & := \lambda a. \text{fold}(in_0 \ a) : A \rightarrow \text{Res}^6\ A \\
\text{Cont}^6 & := \lambda k. \text{fold}(in_1 \ k) : (I \rightarrow (0 \times \text{Res}^6\ A)) \rightarrow \text{Res}^6\ A
\end{align*}
\]

\( \text{Ret}^6(a) \) represents a computation which returns the value \( a : A \), while \( \text{Cont}^6(f, k) \) (with \( (f, k) : I \rightarrow (0 \times \text{Res}^6\ A) \)) represents a computation which on input \( i : I \) outputs \( f_i : 0 \) and continues with \( k_i : \text{Res}^6\ A \). Given \( p, q : \text{Res}^6\ A \), the scheduler \( \text{(sched} \ p \ q) \), adapted from \([18]\), first evaluates \( p \) if \( p \) returns, then the whole computation returns, with the same value. Otherwise, \( p \) evaluates to say \( \text{Cont}^6(f, k) \). Then \( \text{(sched} \ p \ q) \) produces a computation which on input \( i : I \) outputs \( f_i \) and continues with \( \text{(sched} \ q \ (k_i)) \), thus switching arguments.

Let \( I \) be a finite base type (so that \( \text{Res}^6\ A \) is finitary polynomial). Let \( \psi : A, \vartheta : 0 \) and \( \varphi : \text{Res}^6\ A \). We have the following formulae (where \( i : I \):

\[
\begin{align*}
[\text{Ret}] & := [\text{fold}(\text{in}_0 \ a) \upharpoonright I] \\
[\text{now}] & := [\text{fold}(\text{in}_1 \ a) \upharpoonright I] \quad [\text{out}_i] & := [\text{fold}(\text{in}_1 \ a) \upharpoonright \{i\}] \longrightarrow [\pi_0 \vartheta] \\
\bigcirc_1 \varphi & := [\text{fold}(\text{in}_1 \ a) \upharpoonright I] \longrightarrow [\pi_1 \text{next} \varphi]
\end{align*}
\]

The formula \([\text{Ret}]\) (resp. \([\text{now}] \psi\)) holds on a resumption which immediately returns (resp. with a value satisfying \( \psi \)) and we have \( \text{Ret}^6 : A \rightarrow \{\text{Res}^6\ A \mid [\text{Ret}]\} \), \( \text{Cont}^6 : \{I \rightarrow (0 \times \text{Res}^6\ A) \mid [\text{out}_i]\} \rightarrow \{\text{Res}^6\ A \mid [\text{out}_i]\} \). Moreover, the typings

\[
\begin{align*}
\text{Cont}^6 & := \{I \rightarrow (0 \times \text{Res}^6\ A) \mid [\text{out}_i]\} \longrightarrow \{\text{Res}^6\ A \mid [\text{out}_i]\} \\
\text{Cont}^6 & := \{I \rightarrow (0 \times \text{Res}^6\ A) \mid [\text{out}_i]\} \longrightarrow \{\text{Res}^6\ A \mid [\text{out}_i]\}
\end{align*}
\]

express that \([\text{out}_i] \vartheta : \text{Res}^6\ A \) is satisfied by \( \text{Cont}^6(f, k) \) if \( f_i \) satisfies \( \vartheta \), and that \( \bigcirc_1 \varphi : \text{Res}^6\ A \) is satisfied by \( \text{Cont}^6(f, k) \) if \( k_i \) satisfies \( \text{next} \varphi \). Since \( I \) is a finite base type, it is possible to quantify over its inhabitants. We thus obtain CTL-like variants of \( \Box \) and \( \Diamond \) (Ex. \([\Box.3]\) and Ex. \([\Diamond.3]\)). Namely:

\[
\begin{align*}
[\Box\text{out}] & := \bigwedge_{i \in I} [\text{out}_i] \\
[\Diamond\text{out}] & := \bigvee_{i \in I} [\text{out}_i] \\
[\Box\varphi] & := \Box \varphi := \bigwedge_{i \in I} \bigwedge_{i} \varphi : \text{Res}^6\ A \\
[\Diamond\varphi] & := \Diamond \varphi := \bigvee_{i \in I} \bigvee_{i} \varphi : \text{Res}^6\ A \\
\forall \varphi & := \mu \alpha. \varphi \land \alpha : \text{Res}^6\ A \\
\exists \varphi & := \nu \alpha. \varphi \land \alpha : \text{Res}^6\ A \\
[\text{Ret}] & := [\text{fold}(\text{in}_0 \ a) \upharpoonright I] \\
[\text{now}] & := [\text{fold}(\text{in}_1 \ a) \upharpoonright I] \longrightarrow [\pi_0 \vartheta] \\
[\bigcirc_1 \varphi] & := [\text{fold}(\text{in}_1 \ a) \upharpoonright \{i\}] \longrightarrow [\pi_1 \text{next} \varphi]
\end{align*}
\]

Our system can prove that \( \Gamma[\text{sched}] \) returns in finite time when so do its arguments, either along some or along any sequence of inputs. We moreover have expected \( \Box \Diamond \) properties for all possible (consistent) combinations of \( \exists/\forall \) and \( [\text{Ret}] / [\text{now}] / [\Box\text{out}] \) (Table \([\Box.4]\) with \( \psi : A, \vartheta : 0 \) safe and smooth). See \([E.7]\) \( \Box \)
Example 8.6 (Breadth-First Traversal). The function \( \text{bft} \) of Table 3 (where \( g \) stands for \( \lambda x. g \odot x \)) implements Martin Hofmann’s algorithm for breadth-first tree traversal. This algorithm involves the higher-order type \( \text{Rou}^g A \) (see Ex. 3.2) with constructors \( \text{Over}^g := \text{fold}(\text{in}_0()) : \text{Rou}^g A \) and \( \text{Cont}^g := \lambda f. \text{fold}(\text{in}_1 f) : ([\text{▶} \text{Rou}^g A \rightarrow ▶ A] \rightarrow A) \rightarrow \text{Rou}^g A \)

We refer to [10] for explanations. Consider a formula \( \varphi : A \). We can lift \( \varphi \) to \( [\text{Rou}^g] \varphi := \nu \alpha. [\text{fold}][\text{in}_1](((\text{next}^\alpha \parallel \text{next}^\alpha)) \parallel \varphi) : \text{Rou}^g A \)

We then easily derive the expected refinement type of \( \text{bft}^g \) (Table 4, where \( \vartheta : C \)). Assume that \( \vartheta \) is safe. On the one hand it is not clear what the meaning of \( [\text{Rou}]\vartheta \) is, because it is an unsafe formula over a non-polynomial type. On the other hand, the type of \( \text{bft}^g \) in Tab. 4 has its standard expected meaning (namely: if all nodes of a tree satisfy \( \vartheta \) then so do all elements of its traversal) because the types \( \text{Tree}^g C, \text{CoList}^g C \) are polynomial and the formulae \( \forall \varphi \exists \vartheta \) are safe. Hence, our system can prove standard statements via detours through non-standard ones, which illustrates its compositionality. We have the same typing for a usual breadth-first tree traversal with forests (à la [39]). See [E.8] ⊓ ⊔

9 Related Work

Type systems based on guarded recursion have been designed to enforce properties of programs handling coinductive types, like causality [49], productivity [51,55,20,61,28,27], or termination [67]. These properties are captured by the type systems, meaning that all well-typed programs satisfy these properties.

In an initially different line of work, temporal logics have been used as type systems for functional reactive programming (FRP), starting from LTL [35,36] to the intuitionistic modal \( \mu \)-calculus [18]. These works follow the Curry-Howard “proof-as-programs” paradigm, and reflect in the programming languages the constructions of the temporal logic.

The FRP approach has been adapted to guarded recursion, e.g. for the absence of space leaks [15], or the absence of time leaks, with the Fitch-style system of [7]. This more recently lead [8] to consider liveness properties with an FRP approach based on guarded recursion. In this system, the guarded \( \lambda \)-calculus (presented in a Fitch-style type system) is extended with a delay modality (written \( \odot \)) together with a “until type” \( A \text{ Until } B \). Following the Curry-Howard correspondence, \( A \text{ Until } B \) is eliminated with a specific recursor, based on the usual unfolding of Until in LTL, and distinct from the guarded fixpoint operator.

In these Curry-Howard approaches, temporal operators are wired into the structure of types. This means that there is no separation between the program and the proof that it satisfies a given temporal property. Different type formers having different program constructs, different temporal specifications for the same program may lead to different actual code.

We have chosen a different approach, based on refinement types, with which the structure of formulae is not reflected in the structure of types. This allows
for our examples to be mostly written in a usual guarded recursive fashion (see Table 3). Of course, we indeed use the modality ■ at the type level as a separation between safety and liveness properties. But different liveness properties (e.g. ◇, ◇□, □◇) are uniformly handled with the same ■-type, which is moreover the expected one in the guarded λ-calculus [20].

Higher-order model checking (HOMC) [59,43] has been introduced to check automatically that higher-order recursion schemes, a simple form of higher-order programs with finite data-types, satisfy a μ-calculus formula. Automatic verification of higher-order programs with infinite data-types (integers) has been explored for safety [44], termination [50], and more generally ω-regular [56] properties. In presence of infinite datatypes, semi-automatic extensions of HOMC have recently been proposed [46]. In contrast with this paper, most HOMC approaches do not consider input-output behaviors on coalgebraic data. A notable exception is [45,26], but it does not handle higher-order functions (such as map), nor polynomial types such as Str(Str A) (Ex. 8.3) or non-positive types such as Rou A (Ex. 8.6) and imposes a strong linearity constraint on pattern matching.

Event-driven approaches consider effects generating streams of events [66], which can be checked for temporal properties with algorithms based on (HO)MC [30,31], or, in presence of infinite datatypes, with refinement type systems [46,58]. Our iteration terms can be seen as oracles, as required by [46] to handle liveness properties, but we do not know if they allow for the non-regular specifications of [55]. While such approaches can handle infinite data types with good levels of automation, they do not have coinductive types nor branching time properties, such as the temporal specification of sched on resumptions (Ex. 8.5).

Along similar lines, branching was approached via non-determinism in [69], which also handles universal and existential properties on traces. This framework can handle CTL-like properties of the form ∃/∀-□/◇ (with our notation of Ex. 8.5), but not nested combinations of these (as e.g. ∃□◇ for sched in Ex. 8.5). It moreover does not handle coinductive types.

10 Conclusion and Future Work

We have presented a refinement type system for the guarded λ-calculus, with refinements expressing temporal properties stated as (alternation-free) μ-calculus formulae. As we have seen, the system is general enough to prove precise behavioral input/output properties of coinductively-typed programs. Our main contribution is to handle liveness properties in presence of guarded recursive types. As seen in §2, this comes with inherent difficulties. In general, once guarded recursive functions are packed into coinductive ones using ■, the logical reasoning is made in our system directly on top of programs, following their shape, but requiring no further modification. We thus believe to have achieved some separation between programs and proofs.

We provided several examples. While they demonstrate the flexibility of our system, they also show that more abstraction would be welcomed when proving
liveness properties. In addition, our system lacks expressiveness to prove e.g. liveness properties on breadth-first tree traversals.

We believe that our approach could be generalized to other programming languages with inductive or coinductive types. The key requirement are: (1) modalities in the temporal logic to navigate through the types of the languages, (2) a semantics to indicate when a program satisfies a formula of the temporal logic, which is sufficiently closed to the set-theoretic one for liveness properties to get their expected meaning, and (3) inference rules to reason over this realizability semantics.

Extensions of the guarded \( \lambda \)-calculus with dependent types have been explored \[14\,11\,6\,27\]. It may be possible to extend our work to these systems. This would require to work in a Fitch-style presentation of the \( \Box \) modality, as in \[7\,12\], since it is not known how to extend delayed substitutions to dependent types while retaining decidability of type-checking \[15\]. Also, it is appealing to investigate the generalization of our approach to sized types \[1\], in which guarded recursive types are representable \[72\].

We plan to investigate type checking. For instance, in a decidable fragment like the \( \mu \)-calculus on streams, one can check that a function of type \( \{ \text{Str}^\Box C \mid \Diamond \Box \text{hd} \psi \} \rightarrow \{ \text{Str}^\Box B \mid \Diamond \Box \text{hd} \psi \} \) can be postcomposed with one of type \( \{ \text{Str}^\Box B \mid \Box \Diamond \text{hd} \psi \} \rightarrow \{ \text{Str}^\Box A \mid \Box \Diamond \text{hd} \varphi \} \) (since \( \Diamond \Box \text{hd} \psi \Rightarrow \Box \Diamond \text{hd} \psi \)). Hence, we expect that some automation is possible for fragments of our logic. In presence of iteration terms, arithmetic extensions of the \( \mu \)-calculus \[41\,42\] may provide interesting backends. An other direction is the interaction with HOMC. If (say) a stream over \( A \) is representable in a suitable format, one may use HOMC to check whether it can be argument of a function expecting e.g. a stream of type \( \{ \text{Str}^\Box A \mid \Box \Diamond \text{hd} \varphi \} \). This might provide automation for fragments of the guarded \( \lambda \)-calculus. Besides, the combination of refinement types with automatic techniques like predicate abstraction \[62\], abstract interpretation \[37\], or SMT solvers \[71\,70\] has been particularly successful. More recently, the combination of refinement types inference with HOMC has been investigated \[64\].

We would like to explore temporal specification of general, effectful programs. To do so, we wish to develop the treatment of the coinductive resumptions monad \[60\], that provides a general framework to reason on effectful computations, as shown by interaction trees \[75\]. It would be interesting to study temporal specifications we could give to effectful programs encoded in this setting. To formalize reasoning on such examples, we would like to design an embedding of our system in a proof assistant like Coq.

Following \[3\], guarded recursion has been used to abstract the reasoning on step-indexing \[1\] that has been used to design Kripke Logical Relations \[2\] for typed higher-order effectful programming languages. Program logics for reasoning on such logical relations \[21\,22\] uses this representation of step-indexing via guarded recursion. It is also found in Iris \[40\], a framework for higher-order concurrent separation logic. It would be interesting to explore the incorporation of temporal reasoning, especially liveness properties, in such logics.
References


52. Mac Lane, S., Moerdijk, I.: Sheaves in geometry and logic: A first introduction to topos theory. Springer (1992)


A Additional Material for §3 (The Pure Calculus)

The typing rules for our pure calculus (i.e. the guarded λ-calculus of [20]) are given in Fig. 14.

B Additional Material for §4 (A Temporal Modal Logic)

Figure 15 presents the definition of the variance predicates \( \alpha \) Pos \( \varphi \) and \( \alpha \) Neg \( \varphi \) for the full logical language ([14] and [16]). The intuitionistic propositional deduction rules are given in Fig. 16.

Remark B.1. All modalities ([\( \pi \)] \( i \)), [fold], [next], [inX], [ev(\( \psi \))] and [box(\( \psi \))] satisfy the monotonicity rule (RM) and are thus monotone in the sense of [19], from which we borrowed the terminology used in Table 2 (see also [29][25]). Assuming the rule (RM), we easily get the following:

(a) Axiom (N) implies the usual necessitation rule:

$$\varphi \vdash [\Delta] \varphi$$  (RN)

Proof. Indeed, one can derive

$$\frac{\vdash \varphi}{\vdash [\Delta] \varphi}$$  (N)

(b) Axiom (C) implies the usual axiom (K):

$$[\Delta](\varphi \Rightarrow \psi) \iff ([\Delta] \varphi \Rightarrow [\Delta] \psi)$$
Proof. Indeed, one has

\[
\begin{align*}
\text{(RM)} & \quad ((\varphi \Rightarrow \psi) \land \varphi) \Rightarrow \psi \\
\text{(C)} & \quad [\Delta](\varphi \Rightarrow \psi) \land [\Delta] \psi \\
& \quad [\Delta](\varphi \land [\Delta] \psi) \\
& \quad [\Delta](\varphi \land [\Delta] \psi) \Rightarrow [\Delta]([\Delta] \varphi \Rightarrow [\Delta] \psi)
\end{align*}
\]

(c) We have the monotonicity axioms

\[
[\Delta](\varphi \land [\Delta] \psi) \Rightarrow [\Delta] \varphi \land [\Delta] \psi
\]

\[
[\Delta][\Delta] \varphi \lor [\Delta] \psi \Rightarrow [\Delta](\varphi \lor [\Delta] \psi)
\]

In our context, the normal intuitionistic modal logic $IK$ of [61] is $(\text{RM}) + (\text{C}) + (\text{N}) + (\text{P}) + (\text{C}_\psi) + (\text{C}_\psi )$, while the normal modal logic $K$ is $IK + (\text{CL})$ (see e.g. [16]).

C Additional Material for §5 (A Temporally Refined Type System)

The definition of the subtyping relation $\leq$ for the full system ([5] and §6) is given in Fig. 17.
\[
\begin{align*}
\vdash \phi \lor \psi \Rightarrow \phi & \quad \vdash \phi \Rightarrow \phi \lor \psi & \quad \vdash \phi \land \psi \Rightarrow \phi \\
\vdash \phi \lor \psi \Rightarrow \psi \lor \phi & \quad \vdash \phi \land \psi \Rightarrow \psi \land \phi & \quad \vdash \phi \Rightarrow (\psi \Rightarrow \theta) \\
\vdash \phi \Rightarrow \psi & \quad \vdash \psi \Rightarrow \theta & \quad \vdash \theta \lor \phi \Rightarrow \theta \lor \psi
\end{align*}
\]

Fig. 16. Intuitionistic Propositional Deduction Rules.

\[
\begin{align*}
T \leq & T \quad U \leq & V \quad T \leq & U \quad U \leq & T \\
T_0 \leq & U_0 \quad T_1 \leq & U_1 \quad T_0 \leq & T_1 \leq & U_0 \leq & U_1 \\
T_0 \times & T_1 \leq & U_0 \times & U_1 \quad T_0 + & T_1 \leq & U_0 + & U_1 \\
T \leq & [T] & A \leq & \{A \mid \top\} & \{A \mid \phi\} \leq & \{A \mid \psi\} \\
\top \vdash \phi \Rightarrow & \psi & \{B \mid \psi\} \rightarrow & \{A \mid \phi\} \equiv & \{B \rightarrow A \mid \text{ev}(\psi)\} \\
\top \vdash [A \mid \varphi] \equiv & \top \vdash [A \mid \text{next}\varphi] & \forall k \vdash & \top \equiv \top \vdash \forall k \vdash \top \\
\top \vdash \phi & \Rightarrow & \psi & \vdash \phi & \Rightarrow & \psi & \vdash \phi & \Rightarrow & \psi
\end{align*}
\]

Fig. 17. Subtyping Rules (full version).

The underlying pure type \(|T|\) of a refinement type \(T\) is inductively defined as follows:

\[
\begin{align*}
|A| & := A \\
|\{A \mid \varphi\}| & := A \\
|\forall k : T| & := |T| \\
|T + U| & := |T| + |U| \\
|T \times U| & := |T| \times |U| \\
|U \rightarrow T| & := |U| \rightarrow |T|
\end{align*}
\]

D Additional Material for \(\S7\) (Semantics)

This Appendix presents material that we omitted in \(\S7\) for space reasons. We follow roughly the same plan. Most proofs are deferred to App. F. We often use \(\theta\) as a generic notation for \(\mu\) and \(\nu\).
D.1 The Topos of Trees (Basic Structure)

Note D.1. Given an object $X$ of $S$ and $0 < k \leq n$, we write $t \uparrow k$ for the restriction of $t \in X(n)$ into $X(k)$, obtained by composing restriction functions $r^X_i$ for $i = k, \ldots, n - 1$.

Full definitions and proofs of the semantic require the explicit manipulation of some of the structure of $S$. We refer to [13,20] for details.

First, as in any presheaf category, limits and colimits are computed pointwise. In particular binary sums and products are given by

$$(X + Y)(n) = X(n) + Y(n)$$

$$(X \times Y)(n) = X(n) \times Y(n)$$

Moreover, exponentials are induced by the Yoneda Lemma (see e.g. [52, §I.6]). Explicitly, given $S$ object $X$ and $Y$, the exponent $Y^X$ at $n$ is the set of all sequences $(f_\ell)_{\ell \leq n}$ of functions $f_\ell : X(\ell) \rightarrow Y(\ell)$ which are compatible with restriction (i.e. $r^Y_\ell \circ f_{\ell+1} = f_\ell \circ r^X_\ell$).

The morphism $\text{fix}^X : X^\bullet^X \rightarrow X$ is defined as

$$\text{fix}^X_n ((f_m)_{m \leq n}) := (f_n \circ \cdots \circ f_1)(\bullet)$$

The morphism $\text{fix}^X : X^\bullet^X \rightarrow X$ is natural in $X$. Given $f : X \times Y \rightarrow X$ with exponential transpose $f^t : Y \rightarrow X^\bullet^X$, the morphism $\text{fix}^X \circ f^t : Y \rightarrow X$ is unique such that $\text{fix}^X \circ f^t = f \circ (\text{next}^X \circ \text{fix}^X \circ f^t, \text{id}^X)$ ([13 Thm. 2.4]).

Since we do not require the explicit constructions, we refer to [13] for the interpretation of guarded recursive types $\text{Fix}(X).A(X)$ and for the definition of the isos

$$\text{fold} : [A(\text{Fix}(X).A(X))] \rightarrow [\text{Fix}(X).A(X)]$$

$$\text{unfold} : [\text{Fix}(X).A(X)] \rightarrow [A(\text{Fix}(X).A(X))]$$

We now have all the structure we need for the denotational semantics of the □-free fragment of the pure calculus.

D.2 Global Sections and Constant Objects

As for any presheaf topos, the global section functor $\Gamma : S \rightarrow \text{Set}$ is right adjoint to the constant object functor $\Delta : \text{Set} \rightarrow S$ (see e.g. [52, §I.6]):

$$S \xrightarrow{\Gamma} \text{Set} \xleftarrow{\Delta}$$

We record the following easy well-known facts for later use.

Lemma D.2. Given a set $S$ and given $X, Y$ objects of $S$, we have in $\text{Set}$:

1. the unit $\eta : \text{Id}_{\text{Set}} \rightarrow \Gamma \Delta \circ \Delta \Rightarrow \Gamma$ is an iso,
(2) \( \Gamma(X \times Y) \simeq \Gamma X \times \Gamma Y \) and \( \Gamma 1 \simeq 1 \)
(3) \( \Gamma(X + Y) \simeq \Gamma X + \Gamma Y \)
(4) \( \Gamma(X^{\Delta S}) \simeq (\Gamma X)^S \)
(5) \( \Gamma(	ext{next}) \simeq \Gamma X \) (via \( \Gamma(\text{next}) \))

where all the mentioned isos are natural in \( X \) and \( Y \) (when applicable).

Proof.

(1) The unit \( \eta_S \) of \( \Delta \vdash \Gamma \) at \( S \) takes \( a \in S \) to the constant map \((n \mapsto (\bullet \mapsto a)) \in S[1, \Delta S] \). Its inverse is the function \( S[1, \Delta S] \to S \) taking a constant map \( x \in S[1, \Delta S] \) to \( x(0)(\bullet) \).

(2) Since \( \Gamma \) is a right adjoint.

(3) Since for any \( x \in S[1, X + Y] \) there is some \( i \in \{0, 1\} \) such that \( x(\bullet)(n) \) is of the form \( \text{in}_i(x_n) \) for all \( n \in \mathbb{N} \).

(4) Using the Cartesian closed structure of \( S \) and the adjunction \( \Delta \vdash \Gamma \) we have

\[
\Gamma(X^{\Delta S}) = S[1, X^{\Delta S}] \\
\simeq S[1 \times \Delta S, X] \\
\simeq S[\Delta S, X] \\
\simeq \text{Set}[S, \Gamma X]
\]

(5) We show that \( x \in \Gamma X \mapsto \text{next} \circ x \in \Gamma(\text{next}) \) is a bijection. We first show surjectivity. Consider \( x' \in S[1, \text{next}] \). Then for each \( n \in \mathbb{N} \), we have \( x'_{n+1}(\bullet) \in \text{next} X(n+1) = X(n) \) with \( x'_{n+2}(\bullet) = x'_{n+1}(\bullet) \). This defines a map \( x \in S[1, X] \) as \( x_n(\bullet) := x'_{n+1}(\bullet) \). Moreover, \((\text{next}_0 \circ x_n)(\bullet) = \bullet = x'_0(\bullet) \) and

\[
(\text{next}_{n+1} \circ x_{n+1})(\bullet) = x_{n+1}(\bullet) = x'_{n+2}(\bullet) = x'_{n+1}(\bullet)
\]

We now show injectivity. Let \( x, y \in S[1, X] \) and assume \( \text{next} \circ x = \text{next} \circ y : 1 \to S \). Then for all \( n \) we have \( x_{n+1}(\bullet) = y_{n+1}(\bullet) \) and thus \( x_n(\bullet) = y_n(\bullet) \). \( \square \)

Following [20], for a (closed) pure type \( A \), we have

\[
\llbracket A \rrbracket := \Delta \Gamma \llbracket A \rrbracket
\]

In words, all components \( \llbracket A \rrbracket(n) \) are equal to \( \Gamma \llbracket A \rrbracket \), and the restriction maps of \( \llbracket A \rrbracket \) are identities. In particular, a global section \( x \in \Gamma \llbracket A \rrbracket \) is a constant family \((x_n)_{n>0} \) describing a unique global section \( x_{n+1}(\bullet) = x_n(\bullet) \in \Gamma \llbracket A \rrbracket \).

The term constructor \( \text{unbox}(\cdot) \) is interpreted as the counit \( \varepsilon \) of the adjunction \( \Delta \vdash \Gamma \): given \( \varepsilon : M : \llbracket A \rrbracket \), we let \( \llbracket \text{unbox}(M) \rrbracket \) be the composite

\[
\llbracket \varepsilon \rrbracket \llbracket M \rrbracket \llbracket A \rrbracket = \Delta \Gamma \llbracket A \rrbracket \xrightarrow{\varepsilon} \llbracket A \rrbracket
\]

The term constructors \( \text{box} \) and \( \text{prev} \) rely on a strong semantic property of constant types, namely that their interpretation lie (modulo isomorphism) in the image of the constant object functor \( \Delta \).
Definition D.3 ([20 Def. 2.2]). An object $X$ of $S$ is constant if $X \simeq \Delta S$ for some set $S$.

Note that the restriction maps of constant objects are bijections. Similarly as in [20 Def. 2.2], if $x \in X(n)$ with $X$ constant, then we write $x \in X(k)$ for the unique element of $X(k)$ which is equal to $x$ modulo the bijective restriction maps of $X$.

Lemma D.4 ([20 Lem. 2.6]). If $A$ is a constant (pure) type, then $[A]$ is a constant object of $S$.

We now give the interpretations of $\Box_\sigma(M)$ and $\text{prev}_\sigma(M)$ (where $\sigma$ stands for $[x_1 \mapsto M_1, \ldots, x_k \mapsto M_k]$). Assuming in both cases $[M]$ to be defined, for $n > 0$ we let

$$[\Box_\sigma(M)](n) : [\mathcal{E}](n) \rightarrow \Delta \mathcal{E}[A](n) = \mathcal{E}[A]$$

$$\gamma \mapsto \left( m \mapsto [M]_n([M_1]_n(\gamma), \ldots, [M_k]_n(\gamma)) \right)$$

$$[\text{prev}_\sigma(M)](n) : [\mathcal{E}](n) \rightarrow \mathcal{E}[A](n) = [A](n+1)$$

$$\gamma \mapsto \left( [M]_{n+1}([M_1]_n(\gamma), \ldots, [M_k]_n(\gamma)) \right)$$

where the mismatches between $n$ and $m$ and between $n$ and $n + 1$ are legal since $[A_1], \ldots, [A_k]$ are constant by Lem. D.3.

D.3 External and Internal Semantics: Global Definitions

We can now give the full $\mathbf{Set}$ and $\mathcal{S}$ interpretations of the logical language. In both cases, for $\alpha : A \vdash \varphi : A(\alpha)$, we let

$$\varphi^0(\top) := \top$$
$$\varphi^{m+1}(\top) := \varphi(\varphi^m(\top))$$
$$\varphi^0(\bot) := \bot$$
$$\varphi^{m+1}(\bot) := \varphi(\varphi^m(\bot))$$

(Recall that $\theta^\alpha \varphi$ is only allowed when $\varphi$ as at most $\alpha$ as free variable.)

Definition D.5 (External Semantics). Consider a formula $\alpha_1 : A_1, \ldots, \alpha_k : A_k \vdash \varphi : A$ without free iteration variable. Assume given a valuation $v$ taking each propositional variable $\alpha_i$ for $i = 1, \ldots, k$ to a set $v(\alpha_i) \in \mathcal{P}([\Gamma][A_i])$. We define $[\varphi]^A_v \in \mathcal{P}([\Gamma][A])$ by induction on $\varphi$ in Fig. 18.

As for the internal $\mathcal{S}$ semantics $[-]$, we give a global definition, in a form similar to Def. D.5.

Definition D.6 (Internal Semantics). Consider a formula $\alpha_1 : A_1, \ldots, \alpha_k : A_k \vdash \varphi : A$ without free iteration variable. Assume given a valuation $v$ taking each propositional variable $\alpha_i$ for $i = 1, \ldots, k$ to a subobject $v(\alpha_i)$ of $[A_i]$. The subobject $[\varphi]^A_v$ of $[A]$ is defined by induction on $\varphi$ in Fig. 19.

The correctness of Def. D.6, namely that we indeed have $[\varphi]^A_v \in \text{Sub}([A])$, as well as the correspondence with the presentation of [7] are discussed in App. D.6.
{[\bot]}^A_v := 0 \quad {[\top]}^A_v := [A] \quad {[\alpha_i]}^A_v := v(\alpha_i)
{[\varphi \lor \psi]}^A_v := {[\varphi]}^A_v \cup {[\psi]}^A_v \quad {[\varphi \land \psi]}^A_v := {[\varphi]}^A_v \cap {[\psi]}^A_v
{[\psi \Rightarrow \varphi]}^A_v := ([A] \setminus {[\psi]}^A_v) \cup {[\varphi]}^A_v
{[[\pi_i]\varphi]}^{A_0 \times A_1}_v := \left\{ x \in [\pi_v][A_0 \times A_1] \mid \pi_i \circ x \in {[\varphi]}^{A_1}_v \right\}
{[[\text{in}_i]\varphi]}^{A_0 + A_1}_v := \left\{ x \in [\text{in}_v][A_0 + A_1] \mid \exists y \in [\text{in}_v][A_1](x = \text{in}_i \circ y \text{ and } y \in {[\varphi]}^{A_1}_v) \right\}
{[[\text{fold}][\varphi]]^\text{Fix}(X).A}_v := \left\{ x \in [\text{fold}][\varphi]^\text{Fix}(X).A \mid \text{unfold} \circ x \in {[\varphi]}^{A|^\text{Fix}(X).A/X}_v \right\}
{[[\text{ev}(\psi)][\varphi]]^B_v := \left\{ x \in [\text{ev}(\psi)][B \rightarrow A] \mid \forall y \in [\text{ev}][B], y \in {[\varphi]}^B_v \implies \text{ev} \circ (x, y) \in {[\varphi]}^A_v \right\}
{[[\text{box}][\varphi]]^A_v := \left\{ x \in [\text{box}][A] \mid x_1(\bullet) \in {[\varphi]}^A_v \right\}
{[[\text{next}][\varphi]]^A_v := \left\{ \text{next} \circ x \in [\text{next}][A] \mid x \in {[\varphi]}^A_v \right\}
{[[\nu^\alpha \varphi(\alpha)]]^A_v := {[\nu^\alpha \varphi(\alpha)]}^A_v \quad ([\text{fix} = m]) \quad \left\{ x \in [\text{fix}(\alpha)] \mid x \in {[\nu^\alpha \varphi(\alpha)]}^A_v \right\}
{[[\mu^\alpha \varphi(\alpha)]]^A_v := \bigcup \left\{ S \mid S \in \mathcal{P}([\varphi]^A_v) \text{ and } S \subseteq {[\varphi]}^A_v \right\}
{[[\mu^\alpha \varphi(\alpha)]]^A_v := \bigcap \left\{ S \mid S \in \mathcal{P}([\varphi]^A_v) \text{ and } {[\varphi]}^A_v \subseteq S \right\}

Fig. 18. External Semantics.

{[[\bot]]^n_v := 0 \quad {[\top]}^n_v := [A] \quad {[\alpha_i]}^n_v := v(\alpha_i)
{[\varphi \lor \psi]}^n_v := {[\varphi]}^n_v \cup {[\psi]}^n_v \quad {[\varphi \land \psi]}^n_v := {[\varphi]}^n_v \cap {[\psi]}^n_v
{[\psi \Rightarrow \varphi]}^n_v := \left\{ t \in [A](n) \mid \forall k \leq n, t[t^k] \in {[\psi]}^n_v \implies t[t^k] \in {[\varphi]}^n_v \right\}
{[[\pi_i]\varphi]}^{A_0 \times A_1}_v := \left\{ t \in [\pi_v][A_0 \times A_1] \mid \pi_i(t) \in {[\varphi]}^{A_1}_v \right\}
{[[\text{in}_i]\varphi]}^{A_0 + A_1}_v := \left\{ t \in [\text{in}_v][A_0 + A_1] \mid \exists u \in [\text{in}_v][A_1](t = \text{in}_u \circ u \text{ and } u \in {[\varphi]}^{A_1}_v) \right\}
{[[\text{fold}][\varphi]]^\text{Fix}(X).A}_v := \left\{ t \in [\text{fold}][\varphi]^\text{Fix}(X).A \mid \text{unfold} \circ t \in {[\varphi]}^{A|^\text{Fix}(X).A/X}_v \right\}
{[[\text{ev}(\psi)][\varphi]]^B_v := \left\{ t \in [\text{ev}(\psi)][B \rightarrow A] \mid \forall k \leq n, \forall u \in [B^k], u \in {[\varphi]}^B_v \implies (t[t^k])u \in {[\varphi]}^A_v \right\}
{[[\text{box}][\varphi]]^A_v := \left\{ t \in [\text{box}][A] \mid t \in {[\varphi]}^A_v \right\}
{[[\text{next}][\varphi]]^A_v := \left\{ \text{next} \circ t \in [\text{next}][A] \mid t \in {[\varphi]}^A_v \right\}
{[[\nu^\alpha \varphi(\alpha)]]^n_v := {[\nu^\alpha \varphi(\alpha)]}^n_v \quad ([\text{fix} = m]) \quad \left\{ x \in [\text{fix}(\alpha)] \mid x \in {[\nu^\alpha \varphi(\alpha)]}^n_v \right\}
{[[\mu^\alpha \varphi(\alpha)]]^n_v := \bigcup \left\{ S \mid S \in \mathcal{P}([\varphi]^A_v) \text{ and } S \subseteq {[\varphi]}^A_v \right\}
{[[\mu^\alpha \varphi(\alpha)]]^n_v := \bigcap \left\{ S \mid S \in \mathcal{P}([\varphi]^A_v) \text{ and } {[\varphi]}^A_v \subseteq S \right\}

Fig. 19. Internal Semantics.
Remark D.7. For closed formulae we can rephrase Def. D.6 as \( t \models^A_n \varphi \) if \( t \models^A \varphi \), where the forcing relation \( t \models^A_n \varphi \) is inductively defined as follows.

- \( t \not\models^A \bot \).
- \( t \not\models^A \top \).
- \( t \models^A_n \varphi \lor \psi \) if \( t \models^A \varphi \) or \( t \models^A \psi \).
- \( t \models^A_n \varphi \land \psi \) if \( t \models^A \varphi \) and \( t \models^A \psi \).
- \( t \models^A_n \psi \Rightarrow \varphi \) if for all \( k \leq n \), \( t \uparrow k \models^A \varphi \) whenever \( t \uparrow k \models^A \psi \).
- \( t \models^A_{n} \varphi \Leftrightarrow \psi \) iff for all \( k \leq n \), \( t \models^A \varphi \) and \( t \models^A \psi \).
- \( t \not\models^A_{n+1} \varphi \Leftrightarrow \psi \) iff there is \( u \in \{ A \}_{n}(u) \) such that \( t = \text{in}_{i}(u) \) and \( u \not\models^A_{n} \varphi \).
- \( t \models^A_{n} \varphi \) iff for all \( k \leq n \) and all \( u \in \{ B \}_{k}(k) \), \( t(k)(u) \models^A \varphi \) whenever \( u \models^B \psi \).
- \( t \not\models^A_{n+1} \varphi \Leftrightarrow \psi \).
- \( t \models^A \text{next}[\varphi] \).
- \( t \models^A \text{next}[\varphi] \).
- \( t \models^A \text{box}[\varphi] \).

D.4 An Open Geometric Morphism

Key properties of the internal semantics of \([\text{box}]\) rely on some further facts on the adjunction \( \Delta \dashv \Gamma \). We refer to [52],[38].

The functor \( \Delta : \text{Set} \rightarrow S \) preserves limits (in particular, \( \Delta \dashv \Gamma : S \rightarrow \text{Set} \) is a geometric morphism). It follows that \( \Delta \) preserves monos, so that for each set \( S \) the function

\[
A \in \mathcal{P}(S) \mapsto \Delta A \in \text{Sub}(\Delta S)
\]

is a meet preserving (and thus monotone) map. It is easy to see that this map has a posetal left adjoint

\[
f_1 : \text{Sub}(\Delta S) \rightarrow \mathcal{P}(S)
\]

Proof. A subobject \( A \) of \( \Delta S \) is a family of subsets \( A = (A_n) \) with \( A_n \subseteq S \). Hence we can let \( f_1(A) \in \mathcal{P}(S) \) be the set of all \( a \in S \) such that \( a \in A_n \) for some \( n > 0 \). Then assuming \( f_1(A) \subseteq B \) for some set \( B \in \mathcal{P}(S) \), it follows that if \( a \in A_n \) then \( a \in f_1(A) \subseteq B \) so that \( a \in (\Delta B)_n \) and thus \( A \subseteq \Delta B \). Conversely, if \( A \subseteq \Delta B \), then for every \( a \in f_1(A) \), since \( a \in A_n \) for some \( n > 0 \), we must have \( a \in (\Delta B)_n = B \), so that \( f_1(A) \subseteq B \).

As a consequence, the adjoint pair \( \Delta \dashv \Gamma : S \rightarrow \text{Set} \) is an open geometric morphism (in the sense of [52] Def. IX.6.2), from which it follows that \( \Delta \) induces maps of (complete) Heyting algebras \( \mathcal{P}(S) \rightarrow \text{Sub}(\Delta S) \) (see e.g. [52] Thm. X.3.1 & Lem. X.3.2). We state this for later use.

Lemma D.8. For each set \( S \), the functor \( \Delta \) induces a map of (complete) Heyting algebras \( \mathcal{P}(S) \rightarrow \text{Sub}(\Delta S) \).

This means that the \( \text{Set} \) interpretation \( \llbracket \varphi \rrbracket \in \mathcal{P}(\Gamma[A]) \) can be taken to the subobject \( \Delta \llbracket \varphi \rrbracket \in \text{Sub}(\Delta \Gamma[A]) = \text{Sub}(\Gamma[A]) \) in \( S \) while respecting the usual \( \text{Set} \) semantics of logical connectives. In particular, we can allow the logical theory under \( \text{box} \) to be classical, while the \( S \) semantics imposes the ambient logical theory to be intuitionistic.
D.5 Abstract Modalities

We present here some well-known basic material which will help us proving the correctness of the internal and external semantics.

Definition D.9. Let $\mathcal{C}$ be a category with pullbacks and consider a morphism $k : X \to \mathcal{C} Y$.

- The functor $k^* : \mathcal{C}/Y \to \mathcal{C}/X$ is defined by pullbacks

$$
\begin{array}{c}
A' \\
\downarrow_{k^*(g)} \\
X \\
\downarrow_k \\
Y
\end{array}
$$

- The functor $(\exists k) : \mathcal{C}/X \to \mathcal{C}/Y$ is defined by postcomposition:

$$(g : A \to X) \mapsto (k \circ g : A \to Y)$$

The following is a basic property of toposes.

Lemma D.10 ([52, Thm. IV.7.2]). Let $T$ be a topos and fix a map $k : X \to T Y$. The functor $(\exists k)$ is left adjoint to $k^* : T Y \to T X$. Moreover, $k^*$ has a right adjoint $(\forall k)$ and preserves exponentials, and thus preserves subobjects.

Lemma D.11.

(1) The map $(\exists \text{in}_i) : \text{Set}/S_i \to \text{Set}/(S_0 + S_1)$ induces a map $P(S_i) \to P(S_0 + S_1)$.

(2) The map $(\exists \text{in}_i) : S/X_i \to S/(X_0 + X_1)$ induces a map $\text{Sub}(X_i) \to \text{Sub}(X_0 + X_1)$.

Proof. Since in both cases the morphism $\text{in}_i$ is a mono. □

Lemma D.12. The map $S/X \to S \bowtie X$ taking $g : Y \to X$ to $\bowtie (g) : \bowtie Y \to \bowtie X$ induces a map $\text{Sub}(X) \to \text{Sub}(\bowtie X)$.

Proof. The functor $\bowtie$ preserves limits since it has a left adjoint ([13, §2.1]). It thus follows that $\bowtie$ preserves monos. □

D.6 External and Internal Semantics: Local Definitions

Some key properties of the Set and $\mathcal{S}$ interpretations are easier to get if one goes through a local presentation, as operations on subobject and powerset lattices, similar to that of $\mathcal{J}_K$ in §7. The goal is to pave the way toward the correctness of both semantics:

Lemma D.13 (Lem. 7.2). The following holds w.r.t. the full modal theories of Def. [6.4].

(1) If $\vdash A \varphi$ then $[[\varphi]] = \Gamma[A]$.

(2) If $\vdash A \varphi$ then $[[\varphi]] = [A]$.

The detailed proof of Lem. D.13 is deferred to App. F.1. It relies on the following material.
Internal Semantics We use the material of [D.5] to devise operations on subobject lattices corresponding to our modalities. This formally extends the presentation given in §7.

Definition D.14.

(a) Given $S$-objects $X_0$ and $X_1$, define $[[\pi_i]] : \text{Sub}(X_i) \to \text{Sub}(X_0 \times X_1)$ as $\pi_i^*$, where $\pi_i : X_0 \times X_1 \to S X_i$ is the $i$th projection.

(b) Given $S$-objects $X_0$ and $X_1$, define $[[\text{in}_i]] : \text{Sub}(X_i) \to \text{Sub}(X_0 + X_1)$ as $(\exists \text{in}_i)$, where $\text{in}_i : X_i \to S X_0 + X_1$ is the $i$th injection.

(c) Given a locally contractive functor $T$ on $S$, define $[[\text{fold}}] : \text{Sub}(T(\text{Fix}(T))) \to \text{Sub}(\text{Fix}(T))$ as $\text{unfold}^*$, where we have $\text{unfold} : \text{Fix}(T) \to S T(\text{Fix}(T))$.

(d) Given a $S$-object $X$, define $[[\text{next}}] : \text{Sub}(X) \to \text{Sub}(\text{next}X)$ as $\text{next}$.

(e) Given a set $S$, define $[[\text{box}}] : \mathcal{P}(S) \to \text{Sub}(\Delta S)$ as $\Delta$.

We now discuss the case of $[[\text{ev}(\psi)] \varphi$, which is actually interpreted as a logical predicate, in the categorical generalization of the usual sense discussed in [52 §9.2 & Prop. 9.2.4]. We follow here [52 VI.5].

– First, extending the above discussion, for an object $X$ of $S$, the (Heyting algebra) exponent

$(-) \Rightarrow_X (-) : \text{Sub}(X) \times \text{Sub}(X) \to \text{Sub}(X)$

is given by

$$(A \Rightarrow_X B)(n) = \{ t \in X(n) \mid \forall k \leq n, t \uparrow k \in A(k) \Rightarrow t \uparrow k \in B(k) \}$$

(see e.g. [52 Prop. I.8.5]).

– Second, it follows from Lem. D.10 that for objects $X, Y$ of $S$, taking the pullback of the evaluation map $\text{ev} : X^Y \times Y \to X$ gives a map of subobjects, as in

$$
\begin{array}{ccc}
\text{ev}^*(A) & \longrightarrow & A \\
\downarrow & & \downarrow \\
X^Y \times Y & \text{ev} & X
\end{array}
$$

which in particular preserves limits and colimits.

– Third, in the internal logic of $S$, universal quantification over an object $Y$ w.r.t. a predicate $A \in \text{Sub}(X \times Y)$ is given (again via Lem. D.10) by the right adjoint $\forall_Y := \forall \pi$ to $\pi^*$, where $\pi$ is the projection $X \times Y \to X$ ([52 VI.5, p. 300]). Moreover, via the Kripke-Joyal semantics for a presheaf topos ([52 §VI.7, p. 318]), for $A \in \text{Sub}(X \times Y)$, the presheaf $\forall_Y(A)$ at $n$ is

$$\{ t \in X(n) \mid \forall k \leq n, \forall u \in Y(k), (t \uparrow k, u) \in A \}$$

We therefore let, for each pure types $A$ and $B$,

$$[[\text{ev}(\_)]] : \text{Sub}([B]) \to (\text{Sub}([A]) \to \text{Sub}([B \to A]))$$
take $S' \in \text{Sub}(\mathbb{B})$ to

$$[[\text{ev}(S')]] := S \in \text{Sub}(\mathbb{A}) \mapsto \forall_{\mathbb{B}} \left( \pi^*(S') \Rightarrow \forall_{\mathbb{A}^{n_1 \times \mathbb{B}}} \text{ev}^*(S) \right)$$

where $\pi : \mathbb{X} \times Y \to \mathbb{X}$ is a projection.

Now, note that we actually have

**Lemma D.15.** Consider a formula $\Sigma \vdash \varphi : A$ and $v$ as in Def. [D.6] such that $[[\varphi]]_v \in \text{Sub}(\mathbb{A})$. We have

1. $[[\pi_i]_v \varphi] = [[\pi_i]]([\varphi]_v)$
2. $[[\in_i]_v \varphi] = [[\in_i]]([\varphi]_v)$
3. $[[\text{fold}]_v \varphi] = [[\text{fold}}]]([\varphi]_v)$
4. $[[\text{next}]_v \varphi] = [[\text{next}}]]([\varphi]_v)$
5. $[[\text{box}]_v \varphi] = [[\text{box}}]]([\varphi]_v)$
6. $[[\text{ev}(\psi)]_v \varphi] = [[\text{ev}}]]([\varphi]_v)$ for each $\psi : B$ such that $[[\psi]] \in \text{Sub}(\mathbb{B})$.

**Proof.**

1. Since limits are computed pointwise in presheaves, we have

$$[[\pi_i]_v \varphi] = \{(t, u) \in [A_0 \times A_1](n) \times [\varphi](n) \mid u = \pi_i(t)\}$$

which is clearly in bijection with $[[\pi_i]_v \varphi]_{A_0 \times A_1}(n)$.

2. Trivial.
3. Similar to the case of $[\pi_i]$.
4. Trivial.
5. Trivial.
6. Immediate from the above discussion.

We thus have done almost all the work to obtain the following basic fact.

**Lemma D.16.** Given $\alpha_1 : A_1, \ldots, \alpha_k : A_k \vdash \varphi : A$, and $v$ taking $\alpha_i$ for $i = 1, \ldots, k$ to $v(\alpha_i) \in \text{Sub}(\mathbb{A}_i)$, we have $[[\varphi]]^A_{\mathbb{A}^{n_1 \times \mathbb{B}^k}} \in \text{Sub}(\mathbb{A})$.

**Proof.** The proof is by induction on formulae. The interpretation of the propositional connectives follows the corresponding structures in presheaf toposes [52, Prop. I.8.5]. The cases of the modalities $[\Box]$ follow from the induction hypothesis and Lem. [D.15]. The cases of $[\theta_{\alpha} \varphi]$ simply amount to the fact that for presheaf toposes, subobjects lattices are complete ([52, Prop. I.8.5]). The cases of $[\theta_{a\varphi}$ for $\theta$ an iteration term are trivial.

We now turn to the logical theory. We immediately get from the above:

**Corollary D.17.**

1. The maps $[[\pi_i]]$, $[[\text{fold}}]]$ and $[[\text{box}}]]$ are maps of Heyting algebras.
2. The maps $[[\in_i]]$ preserve $\lor$, $\bot$ and $\land$.
3. The maps $[[\text{next}}]]$ preserve $\land$, $\top$ and $\lor$. 
For each object \( X \) of \( S \) and each fixed \( S \in \text{Sub}(X) \), the map \([\text{ev}(S)]\) preserves \( \land, \top \).

Proof.

1. This directly follows from Lem. D.10 and Lem. D.8.
2. Preservation of \( \lor, \bot \) follows from that fact that \([\text{in}_i]\) is a left adjoint by Lem. D.10. For binary conjunctions, first note that meets in partial orders are given by pullbacks. In a subobject lattice \( \text{Sub}(X_i) \), this can be expressed as

\[
\begin{array}{ccc}
A \land B & \rightarrow & B \\
\downarrow & & \downarrow \\
A & \rightarrow & X_i
\end{array}
\]

(where arrows are inclusions maps). Since \( \text{in}_i : X_i \rightarrow X_0 + X_1 \) is a mono, the following is also a pullback in \( \text{Sub}(X_0 + X_1) \):

\[
\begin{array}{ccc}
A \land B & \rightarrow & B \\
\downarrow & & \downarrow \\
A & \rightarrow & X_i \quad \rightarrow \quad X_0 + X_1
\end{array}
\]

3. Preservation of \( \land, \top \) follows from the fact that \( \mathbf{\uparrow}(\cdot) \) is a right adjoint ([13 §2.1]). As for preservation of \( \lor \), we check the details. Consider an object \( X \) of \( S \) and subobjects \( A, B \in \text{Sub}(X) \). We have to show \( \mathbf{\uparrow}(A \lor B) = \mathbf{\uparrow}(A) \lor \mathbf{\uparrow}(B) \).

But we have

\[
\mathbf{\uparrow}(A \lor B)_0 = 1 = 1 \cup 1 = (\mathbf{\uparrow}(A) \lor \mathbf{\uparrow}(B))_0
\]

and

\[
\mathbf{\uparrow}(A \lor B)_{n+1} = (A \lor B)_n = A_n \cup B_n = (\mathbf{\uparrow}(A) \lor \mathbf{\uparrow}(B))_{n+1}
\]

4. This directly follows from Lem. D.10 via Lem. D.15 and the definition of \([\text{ev}(\cdot)]\). \( \square \)

External Semantics We now turn to operations on powerset lattices for the external semantics.

Definition D.18.

(a) Given sets \( S_0 \) and \( S_1 \), define \( \{[\pi_i]\} : \mathcal{P}(S_i) \rightarrow \mathcal{P}(S_0 \times S_1) \) as \( \pi_i^* \), where \( \pi_i : S_0 \times S_1 \rightarrow S_i \) is the \( i \)th projection.
Given a locally contractive functor \( S : \mathcal{C} \to \mathcal{D} \) and a formula \( \varphi : A \to B \), the following functions are Scott cocontinuous:

\[
\begin{align*}
\llbracket \varphi \rrbracket : \text{Sub}(\llbracket A \rrbracket) &\to \text{Sub}(\llbracket B \rrbracket) \\
\llbracket \varphi \rrbracket^\text{op} : \mathcal{P}(\llbracket A \rrbracket) &\to \mathcal{P}(\llbracket B \rrbracket)
\end{align*}
\]

The key for Lem. D.22 is the usual fact that codirected infs commute with infs and finite sups, in Set as well as in \( \mathcal{S} \). The key case of Prop. 7.3 is that of \( \nu \alpha \varphi(\alpha) : A \). We have

\[
\llbracket \nu \alpha \varphi(\alpha) \rrbracket = \bigwedge_{m \in \mathbb{N}} \llbracket \varphi^m(T) \rrbracket \quad \text{and} \quad \llbracket \nu \alpha \varphi(\alpha) \rrbracket^\text{op} = \bigvee_{m \in \mathbb{N}} \llbracket \varphi^m(T) \rrbracket
\]

Given a global section \( x \in \Gamma[\llbracket \nu \alpha \varphi(\alpha) \rrbracket] \), we have

\[
\forall n > 0, \forall m \in \mathbb{N}, \quad x_{n(x)}(\bullet) \in \llbracket \varphi^m(T) \rrbracket[n]
\]

We then easily conclude \( x \in \llbracket \nu \alpha \varphi(\alpha) \rrbracket \) from \( \llbracket \varphi^m(T) \rrbracket = \Gamma[\varphi^m(T)] \). Note that this relies on the commutation of the universal quantifications over \( n \) and \( m \).

The proofs of Lem. D.21, Lem. D.22 and Prop. 7.3 are deferred to App. F.2.
D.8 The Smooth Fragment

The proof of Lem. 7.4 is deferred to App. F.3.

D.9 Constant Objects, Again

For the adequacy of the typing rules of the term constructors box and prev, we need to generalize Lem. D.4 (§D.2) to refinement types. To this end, it is convenient to extend the notation \( J \) to refinement types.

**Definition D.23.** For \( T \) is a type without free iteration variables, we define \( J^T_K \) by induction as follows:

\[
\begin{align*}
\llbracket A | \varphi \rrbracket & := [\varphi] \\
\llbracket \forall^k \cdot T \rrbracket & := \bigwedge_{n \in \mathbb{N}} [T[n/k]] \\
\llbracket T_0 + T_1 \rrbracket & := [T_0] + [T_1] \\
\llbracket T_0 \times T_1 \rrbracket & := [T_0] \times [T_1] \\
\llbracket U \to T \rrbracket & := [U] \to [T] \\
\llbracket \triangleright T \rrbracket & := \triangleright [T] \\
\llbracket \blacksquare T \rrbracket & := \Delta \Gamma[T]
\end{align*}
\]

We can now extend Lem. D.4. We crucially rely on the fact that \( \Delta \) preserves limits (see e.g. [38, Ex. 4.1.4]).

**Lemma D.24.** If \( T \) is a constant type, then \( \llbracket T \rrbracket \) is a constant object of \( S \).

**Proof.** The proof is by induction on types. The cases of the type constructors +, \( \times \), \( \rightarrow \) are easy and discussed in [20, Lem. 2.6]. In the case of \( \text{Fix}(X).A \), since all occurrences of \( X \) in \( A \) should be guarded by a \( \triangleright \), and since \( \blacksquare \) can only be applied to closed types, it follows that \( X \) cannot occur in \( A \). Then \( \llbracket A \rrbracket \) is constant by induction hypothesis and we are done since \( \llbracket \text{Fix}(X).A \rrbracket \simeq \llbracket A \rrbracket \) in this case. The case of \( \blacksquare T \) is trivial. As for refinement types, we show by induction on \( \vdash \varphi: A \) with \( A \) constant that \( J^\varphi_K \) is a constant object.
Cases of $\top$, $\bot$, $\land$, $\lor$ and $\Rightarrow$.
All these cases follow from (the induction hypothesis and) the fact that $\Delta$ induces maps of Heyting algebras on subobject lattices (Lem. D.8).

Case of $\lbrack \text{box} \rbrack \varphi$.
Trivial, since $\lbrack \text{box} \rbrack \varphi$ is in the image of $\Delta$.

Case of $\lbrack \text{next} \rbrack \varphi$.
This case cannot occur since $A$ is constant.

Case of $\lbrack \text{fold} \rbrack \varphi$.
In this case, we have $A = \text{Fix}(X).B$. Since $X$ is guarded in $B$, it must not occur in $B$, and we have $\lbrack A \rbrack \simeq \lbrack B \rbrack$ via unfold. Moreover $\lbrack B \rbrack$ is constant, with say $\lbrack [\varphi] \rbrack \simeq \Delta \Phi$. Now, $\lbrack \text{fold} \rbrack \varphi$ lies in the pullback diagram

\[
\begin{array}{c}
\text{unfold}^*([\varphi]) \simeq \lbrack \text{fold} \rbrack \varphi \\
\downarrow \quad \pi \\
\lbrack A \rbrack \\
\downarrow \quad \text{unfold} \\
\lbrack B \rbrack \simeq \Delta(S)
\end{array}
\]

Since unfold is an iso, the upper arrow $\pi$ is also an iso, and we are done.

Case of $\lbrack \pi_i \rbrack \varphi$.
We rely on the description of $\lbrack \pi_i \rbrack \varphi$ as $\lbrack \pi_i \rbrack \lbrack \varphi \rbrack$ in §D.6. By induction hypothesis and recalling that $\Delta$ preserves finite products, consider the pullback

\[
\begin{array}{c}
\pi^*([\varphi]) \simeq \lbrack \pi_i \rbrack \varphi \\
\downarrow \\
\Delta(S_0) \times \Delta(S_1) \\
\downarrow \quad \pi_i \\
\Delta(S_i)
\end{array}
\]

Then one can take the corresponding pullback in Set

\[
\begin{array}{c}
\Psi \\
\downarrow \\
S_0 \times S_1 \\
\downarrow \quad \pi_i \\
S_i
\end{array}
\]

and this implies that $\lbrack \pi_i \rbrack \varphi \simeq \Delta(\Psi)$ since $\Delta$ preserves finite limits.

Case of $\lbrack \text{in} \rbrack \varphi$.
We rely on the description of $\lbrack \text{in} \rbrack \varphi$ as $\lbrack \text{in} \rbrack \lbrack \varphi \rbrack$ in §D.6. The result follows from the induction hypothesis and the fact that $\Delta$ preserves finite limits and colimits, as in:

\[
\lbrack \varphi \rbrack \simeq \Delta(\Phi) \hookrightarrow \Delta(S_i) \xrightarrow{\Delta(\text{in})} \Delta(S_0) + \Delta(S_1)
\]

Case of $\lbrack \text{ev}(\psi) \rbrack \varphi$.
We rely on the description of $\lbrack \text{ev}(\psi) \rbrack \varphi$ in §D.6 that is

\[
\lbrack \text{ev}(\psi) \rbrack \varphi = \forall [B] \left( \pi^*([\psi]) \implies \text{in} \lbrack [B] \rbrack \text{ev}^*([\varphi]) \right)
\]
The result then follows from Lem. D.8 and the fact that $\Delta$ thus preserves universal quantifications (see e.g. [52, Thm. X.3.1 & Lem. X.3.2]).

**Cases of $\theta^i\alpha\varphi$ and $\theta\alpha\varphi$.**

By assumption, the occurrences of $\alpha$ in $\varphi$ should be guarded by a $\text{[next]}$.

Since $\text{[box]}$ can only be applied to closed formulae, this imposes $\alpha$ not to appear in $\varphi$. But then the result follows by induction hypothesis. □

### D.10 Realizability

We detail the steps toward the Adequacy Theorem 7.7. Full proofs are deferred to App. F.4. The first basic result we need about our notion of realizability is that it is monotone w.r.t. step indexes.

**Lemma D.25 (Monotonicity of Realizability).** Let $T$ be a type without free iteration variables. If $x \vDash_n U$ then $x \vDash_k U$ for all $k \leq n$.

The correctness of subtyping requires two additional lemmas. The first one concerns the rule

$$\Delta \leq |\Delta|$$

**Lemma D.26.** For a pure type $A$ and $x \in \Gamma[A]$, we have $x \vDash_n A$ for all $n > 0$.

Second, we need a result of [20] for the correctness of the subtyping rules

$$\{A \mid \varphi\} \leq \{\Box A \mid \Box \varphi\}$$

$$\{B \mid \psi\} \Rightarrow \{A \mid \varphi\} \leq \{B \Rightarrow A \mid [\text{ev(}\psi\text{)}] \varphi\}$$

$$\mathcal{E}, x : \{B \mid \psi\} \vdash M : \{A \mid \varphi\}$$

An object $X$ of $S$ is total if all its restriction maps $r^n_X : X_{n+1} \rightarrow X_n$ are surjective. Hence, if $X$ is total, then given $t \in X_n$ for some $n > 0$, there is a global section $x : 1 \rightarrow_S X$ such that $x_n(\bullet) = t$.

**Lemma D.27 ([20, Cor. 3.8]).** For a pure type $A$, the object $[A]$ is total.

We then obtain the correctness of subtyping as usual. The rules

$$\vdash^A \varphi \Rightarrow \psi$$

$$\vdash^A \varphi \Rightarrow \psi$$

rely on Lem. D.13 (Lem. 7.2), while

$$\varphi \text{ safe}$$

$$\Box \{A \mid \varphi\} \equiv \Box \{A \mid \Box \varphi\}$$

is given by Prop. 7.3.

**Lemma D.28 (Correctness of Subtyping (Lem. 7.6)).** Given types $T, U$ without free iteration variable, if $x \vDash_n U$ and $U \leq T$ then $x \vDash_n T$. 
We introduce some notation to manipulate these global sections. Given a typing $M$ say to some $\rho(y)$ such that, if $\rho(x_i)$ in the standard sense (i.e. $\rho$ takes each $x_i$ for $i = 1, \ldots, k$ to some $\rho(x_i) \in \Gamma'[\Gamma[T]]$. Given a typing judgment $\mathcal{E} \vdash M : T$, we let

$$[M] \circ \langle u_1, \ldots, u_k \rangle : 1 \rightarrow \|T\|$$

We introduce some notation to manipulate these global sections. Given a typing context $\mathcal{E} = x_1 : T_1, \ldots, x_k : T_k$ we write $\rho \models \mathcal{E}$ if $\rho$ takes each $x_i$ for $i = 1, \ldots, k$ to some $\rho(x_i) \in \Gamma'[\Gamma[T]]$. Given a typing judgment $\mathcal{E} \vdash M : T$, we let

$$[M]_{\rho} := [M] \circ \langle \rho(x_1), \ldots, \rho(x_k) \rangle$$

Given $\rho \models \mathcal{E}$ and $n > 0$, write $\rho \equiv_n \mathcal{E}$ if $\rho(x_i) \equiv_n T_i$ for all $i = 1, \ldots, k$. Thm. 7.7 is proved under the following form.

**Theorem D.29 (Adequacy (Thm. 7.7)).** Let $\mathcal{E}, T$ have free iteration variables among $\mathcal{E}$, and let $\bar{m} \in \mathbb{N}$. If $\mathcal{E} \vdash M : T$ and $\rho \models \mathcal{E}$, then

$$\forall n > 0, \quad \rho \equiv_n \mathcal{E}[\bar{m}/\bar{m}] \quad \Rightarrow \quad [M]_{\rho} \equiv_n T[\bar{m}/\bar{m}]$$

**Corollary D.30.** (1) Consider a closed term $\vdash M : \{A \mid \varphi\}$ with $\varphi$ safe. Then $\llbracket M \rrbracket : 1 \rightarrow_{\mathcal{S}} \llbracket A \rrbracket \in \llbracket \varphi \rrbracket$.

(2) Consider a closed term $\vdash M : \{A \mid \psi\} \rightarrow \{A \mid \varphi\}$, with $\varphi$, $\psi$ safe. Then $\llbracket M \rrbracket$ induces a function $\Gamma[M]$ taking $x \in \llbracket \psi \rrbracket$ to $\Gamma[M] = [M] \circ x \in \llbracket \varphi \rrbracket$.

Corollary D.30 of course extends to any arity. As a consequence of Cor. D.30 and Mogelberg’s Theorem 7.1 [55], for a closed term $M : \{\llbracket P \mid [\boxtimes] \varphi\}$ with $P$ polynomial, the unique global section $[M]_{n+1}(\bullet) \in \llbracket P \rrbracket$ satisfies $\varphi$ in the standard sense (i.e. $[M]_n(\bullet) \in \llbracket \varphi \rrbracket$). Moreover a function, say $M : \{\llbracket Q \mid [\boxtimes] \psi\} \rightarrow \{\llbracket P \mid [\boxtimes] \varphi\}$ with $Q, P$ polynomial induces a Set-function

$$\Gamma[M] : \Gamma[\llbracket Q \rrbracket] \rightarrow \Gamma[\llbracket P \rrbracket]$$

such that, if $y \in \Gamma[\llbracket Q \rrbracket] \simeq \Gamma[\Delta][\llbracket Q \rrbracket] = \Gamma[\llbracket Q \rrbracket]$ satisfies $\psi$ in the standard sense (i.e. $y \in \llbracket \varphi \rrbracket$), then the unique global section $\Gamma[M](y)_{n+1}(\bullet) = \Gamma[M](y)_n(\bullet) \in \Gamma[P]$ satisfies $\varphi$ in the standard sense (i.e. belongs to $\llbracket \varphi \rrbracket$).

### D.11 A Galois Connection

It is common for the classification of temporal properties to identify safety properties with topologically closed sets and to identify liveness properties with topologically dense sets. As any subset of a topological space is the intersection of a closed set with a dense set, this provides a topological decomposition of temporal properties, which furthermore restricts to regular properties on (finitary) polynomial types. We refer to e.g. [2].
Here, we make explicit the relation between safe formulae on polynomial types (in the sense of Def. 6.5) and safety properties understood as closed subsets of the corresponding final Set-coalgebras (in view of Mogelberg’s Theorem [55]), for the usual tree (or stream) topology.

First, it might be useful to remember what it means for a global section $x \in \Gamma X$ in $\mathcal{S}$ to satisfy a property $S$, where $S \in \text{Sub}(X)$ is a subobject of $X$. Following e.g. [52,51], we say that $x \in \Gamma X$ satisfies a property $S \in \text{Sub}(X)$ if $x$ factors through $S$, as in

$$1 \xrightarrow{x} S \xrightarrow{} X$$

that is: $\forall n > 0, \ x_n(\bullet) \in S(n)$

Fix an object $X$ of $\mathcal{S}$. There is a Galois connection between the subobjects of $X$ in $\mathcal{S}$ and the subsets of $\Gamma X$ in $\text{Set}$:

$$\text{Pref} \dashv \text{Clos} : \text{Sub}(X) \to \mathcal{P}(\Gamma X)$$

where for $S \in \mathcal{P}(\Gamma X)$ and $B \in \text{Sub}(X)$,

$$\text{Pref}(S) : n \mapsto \{x_n(\bullet) \mid x \in S\}$$
$$\text{Clos}(B) := \{x \in \Gamma X \mid \forall n > 0, \ x_n(\bullet) \in B(n)\}$$

Of course, Clos is the restriction of $\Gamma : \mathcal{S} \to \text{Set}$ to the subobjects of $X$.

Let us spell out the fact that $\text{Pref} \dashv \text{Clos}$ form a Galois connection. Fix an object $X$ of $\mathcal{S}$. First, it is trivial that the functions

$$\text{Pref} : \mathcal{P}(\Gamma X) \to \text{Sub}(X)$$
$$\text{Clos} : \text{Sub}(X) \to \mathcal{P}(\Gamma X)$$

are monotone w.r.t. the orders of the lattices $\mathcal{P}(\Gamma X)$ and $\text{Sub}(X)$. Moreover, we have:

**Lemma D.31.** We have

(i) $S \subseteq \text{Clos}(\text{Pref}(S))$ for $S \in \mathcal{P}(\Gamma X)$.

(ii) $\text{Pref}(\text{Clos}(B)) \subseteq B$ for $B \in \text{Sub}(X)$.

**Proof.**

(i) Given $x \in S$, by definition we have $x_n(\bullet) \in \text{Pref}(S)(n)$ for all $n > 0$, so $x \in \text{Clos}(\text{Pref}(S))$.

(ii) Given $a \in \text{Pref}(\text{Clos}(B))(n)$, there is some $x \in \text{Clos}(B)$ such that $a = x_n(\bullet)$. But $x \in \text{Clos}(B)$ means $x_k(\bullet) \in B(k)$ for all $k > 0$, so that $a = x_n(\bullet) \in B(n)$.

As usual, we trivially get

$$\text{Pref}(S) \leq B \quad \text{iff} \quad S \subseteq \text{Clos}(B)$$
Say that \( S \in \mathcal{P}(\Gamma X) \) is closed if \( S = \text{Clos}(B) \) for some \( B \in \text{Sub}(X) \). It is easy to see that \( S \) is closed if and only if \( S = \text{Clos}(\text{Pref}(S)) \). Note that \( S = \text{Clos}(\text{Pref}(S)) \) unfolds to

\[
\forall x \in \Gamma [A], \quad x \in S \iff \forall n > 0, \ \exists y \in S, \ x_n(\bullet) = y_n(\bullet)
\]

When \( A \) is a polynomial recursive type, Thm. 7.1 thus says that \( S \) is closed if and only if \( S \) is closed for the corresponding usual tree (or stream) topology. Since Prop. 7.3 can be formulated as

\[
\{\|\varphi\|\} = \text{Clos}(\|\varphi\|)
\]

it indeed says that \( \{\|\varphi\|\} \) is closed for the usual topology.

We finally briefly elaborate on this in view of the coincidence of the \( S \) and \textbf{Set} semantics for safe formulae (Prop. 7.3). Let us consider the cases of \( \square[\text{hd}]\varphi \) and \( \Diamond[\text{hd}]\varphi \) on guarded streams \( \text{Str}^k \mathcal{B} \). Assume that \( \varphi \) is safe. The equality \( \{\square[\text{hd}]\varphi\} = \Gamma[\square[\text{hd}]\varphi] \) implies that the usual \textbf{Set} semantics of \( \square[\text{hd}]\varphi \) is in the image of \( \Gamma \). But a subset of \( \Gamma[\text{Str}^k \mathcal{B}] \) which is in the image of \( \Gamma \) is necessarily a closed set w.r.t. the usual product topology on streams in \textbf{Set}, i.e. a safety property. Formulae of the form \( \square[\text{hd}]\varphi \) define safety properties on streams, but liveness properties of the form \( \Diamond[\text{hd}]\varphi \) are not closed (for non-trivial \( \varphi \)), and thus cannot be in the image of \( \Gamma \).
E Details of the Examples

E.1 Guarded Streams

The Later Modality on Guarded Streams

Example E.1. We have the following basic modal refinement types for Cons and tl:

\[
\begin{align*}
\text{Cons}^\# &: A \rightarrow \triangleleft \{ \text{Str}^\# A \mid \varphi \} \rightarrow \{ \text{Str}^\# A \mid \bigcirc \varphi \} \\
\text{tl}^\# &: \{ \text{Str}^\# A \mid \bigcirc \varphi \} \rightarrow \triangleleft \{ \text{Str}^\# A \mid \varphi \}
\end{align*}
\]

Proof. We begin with Cons. Recall that Cons = \( \lambda x. \lambda s. \text{fold}(x, s) \) and that \( \bigcirc(-) = [\text{fold}]|\pi_1|\text{next}(-) \). The result then follows from the following derivation:

\[
\begin{align*}
x : A, s : \triangleleft \{ \text{Str}^\# A \mid \varphi \} & \vdash s : \triangleleft \{ \text{Str}^\# A \mid \varphi \} \\
\vdash s : \triangleleft \{ \text{Str}^\# A \mid \varphi \} & \vdash \text{unfold} s : A \times \triangleleft \{ \text{Str}^\# A \mid \pi_1|\text{next}|\varphi \} \\
\vdash s : \{ \text{Str}^\# A \mid \varphi \} & \vdash \pi_1(\text{unfold } s) : \triangleleft \{ \text{Str}^\# A \mid \varphi \}
\end{align*}
\]

As for tl, recalling that tl = \( \lambda s. \pi_1(\text{unfold } s) \), the result follows from

\[
\begin{align*}
\vdash s : \{ \text{Str}^\# A \mid \varphi \} & \vdash s : \{ \text{Str}^\# A \mid [\text{fold}]|\pi_1|\text{next}|\varphi \} \\
\vdash s : \{ \text{Str}^\# A \mid \varphi \} & \vdash \text{unfold } s : A \times \triangleleft \{ \text{Str}^\# A \mid \pi_1|\text{next}|\varphi \} \\
\vdash s : \{ \text{Str}^\# A \mid \varphi \} & \vdash \pi_1(\text{unfold } s) : \triangleleft \{ \text{Str}^\# A \mid \varphi \}
\end{align*}
\]

□

Destructor of Guarded Streams

Example E.2. The types of hd and tl can be refined as follows with the always modality \( \Box \):

\[
\begin{align*}
\text{hd}^\# &: \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} \rightarrow \{ A \mid \varphi \} \\
\text{tl}^\# &: \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} \rightarrow \triangleleft \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \}
\end{align*}
\]

Proof. Recall that \( [\text{hd}]\varphi = [\text{fold}]|\pi_0|\varphi \). We begin with the typing of

\[
\text{hd}^\# = \lambda s. \pi_0(\text{unfold } s) : \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} \rightarrow \{ A \mid \varphi \}
\]

We use \( \vdash \text{Str}^\# A \Box [\text{hd}]\varphi \Rightarrow [\text{hd}]\varphi \).

\[
\begin{align*}
\vdash s : \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} & \vdash s : \{ \text{Str}^\# A \mid [\text{hd}]\varphi \} \\
\vdash s : \{ \text{Str}^\# A \mid [\text{hd}]\varphi \} & \vdash s : \{ \text{Str}^\# A \mid [\text{hd}]\varphi \}
\end{align*}
\]

\[
\begin{align*}
\vdash s : \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} & \vdash \text{unfold } s : A \times \triangleleft \{ \text{Str}^\# A \mid [\pi_0]|\varphi \} \\
\vdash s : \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} & \vdash s : \{ \text{Str}^\# A \mid [\text{hd}]\varphi \}
\end{align*}
\]

\[
\begin{align*}
\vdash \lambda s. \pi_0(\text{unfold } s) : \{ \text{Str}^\# A \mid \Box [\text{hd}]\varphi \} & \rightarrow \{ A \mid \varphi \}
\end{align*}
\]
We continue with the typing of 
\[ t\mathbf{t} := \lambda s.\pi_1(\text{unfold } s) : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \longrightarrow \triangleright \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \]

We use \( \vdash \text{Str}^E A \Box[\text{hd}]\varphi \Rightarrow \Box[\Box[\text{hd}]\varphi] \). Recall that \( \Box[\varphi] = [\text{fold}][\pi_1][\text{next}]\varphi \).

\[
\begin{array}{ccc}
\text{Str}^E A \Box[\text{hd}]\varphi \Rightarrow \Box[\Box[\text{hd}]\varphi] \\
\hline
s : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \vdash s : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} & (\text{Str}^E A \Box[\text{hd}]\varphi \leq \text{Str}^E A \Box[\Box[\text{hd}]\varphi])
\end{array}
\]

\[
\begin{array}{c}
s : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \vdash \text{unfold } s : \{A \times \triangleright \text{Str}^E A \mid [\pi_0][\text{next}]\Box[\text{hd}]\varphi\}
\end{array}
\]

\[
\begin{array}{c}
s : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \vdash \pi_1(\text{unfold } s) : \{\triangleright \text{Str}^E A \mid [\text{next}]\Box[\text{hd}]\varphi\}
\end{array}
\]

\[
\begin{array}{c}
s : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \vdash \pi_1(\text{unfold } s) : \{\triangleright \text{Str}^E A \mid \Box[\text{hd}]\varphi\}
\end{array}
\]

\[
\begin{array}{c}
\vdash \lambda s.\pi_1(\text{unfold } s) : \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \longrightarrow \triangleright \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\}
\end{array}
\]

\[\square\]

**Constructor of Guarded Streams**

**Example E.3.** The type of \( \text{Cons}^E \) can be refined as follows with the **always modality** \( \Box \):

\[ \text{Cons}^E : \{A \mid \varphi\} \longrightarrow \triangleright \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \rightarrow \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \]

**Proof.** We show

\[ \text{Cons}^E := \lambda x.\lambda s.\text{fold}(x, s) : \{A \mid \varphi\} \longrightarrow \triangleright \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \rightarrow \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \]

To this end, we use the following derived rule (see Ex. 5.1):

\[
\begin{array}{c}
\mathcal{E} \vdash M : \{A \mid \varphi\} \\
\mathcal{E} \vdash N : \{B \mid \psi\} \\
\mathcal{E} \vdash (M, N) : \{A \times B \mid [\pi_0]\varphi \land [\pi_1]\psi\}
\end{array}
\]

Consider the typing context

\[ \mathcal{E} := x : \{A \mid \varphi\}, \ s : \triangleright \{\text{Str}^E A \mid \Box[\text{hd}]\varphi\} \]

We know from \[\mathcal{E}.1\] that

\[ \mathcal{E} \vdash \text{fold}(x, s) : \{\text{Str}^E A \mid \Box[\Box[\text{hd}]\varphi]\} \]

Since \( \vdash \text{Str}^E A ([\text{hd}]\varphi \land \Box[\Box[\text{hd}]\varphi]) \Rightarrow \Box[\text{hd}]\varphi \), we are done if we show

\[ \mathcal{E} \vdash \text{fold}(x, s) : \{\text{Str}^E A \mid [\text{hd}]\varphi\} \]

But this is trivial:

\[
\begin{array}{c}
\mathcal{E} \vdash x : \{A \mid \varphi\} \\
\mathcal{E} \vdash (x, s) : \{A \times \triangleright \text{Str}^E A \mid \pi_0]\varphi\} \\
\mathcal{E} \vdash \text{fold}(x, s) : \{\text{Str}^E A \mid [\text{fold}][\pi_0]\varphi\}
\end{array}
\]

\[\square\]
Map over Guarded Streams

*Example E.4.* We have the following:

\[ \text{map}_E : (\{A \mid \varphi\} \to \{B \mid \psi\}) \rightarrow \{\text{Str}_E A \mid \Box[\varphi]\} \rightarrow \{\text{Str}_E B \mid \Box[\psi]\} \]

\[ := \lambda f.\text{fix}(g).\lambda s. f(\text{hd}_E s) :\text{\textit{E}} (g \circ (\text{tl}_E s)) \]

*Proof.* We proceed as follows, using \[E.1\] and \[E.1\]

\[
\begin{align*}
E & : s : \{A \mid \varphi\} \quad E & : s : \{\text{Str}_E A \mid \Box[\varphi]\} \\
E & : \text{hd}_E s : \{A \mid \varphi\} \quad E & : \text{tl}_E s : \Box[\varphi]\}
\end{align*}
\]

where

\[
T := (\{A \mid \varphi\} \to \{B \mid \psi\}) \rightarrow \{\text{Str}_E A \mid \Box[\varphi]\} \rightarrow \{\text{Str}_E B \mid \Box[\psi]\}
\]

\[
E := f : \{A \mid \varphi\} \to \{B \mid \psi\}, g : \Box(\{\text{Str}_E A \mid \Box[\varphi]\} \to \{\text{Str}_E B \mid \Box[\psi]\}), s : \{\text{Str}_E A \mid \Box[\varphi]\}
\]

\[ \square \]

Merge over Guarded Streams

*Example E.5.* We have the following:

\[ \text{merge}_E : \{\text{Str}_E A \mid \Box[\varphi_0]\} \to \{\text{Str}_E A \mid \Box[\varphi_1]\} \rightarrow \{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\} \]

\[ := \text{fix}(g).\lambda s_0.\lambda s_1.\text{Cons}_E (\text{hd}_E s_0) (\text{next}(\text{Cons}_E (\text{hd}_E s_1) (g \circ (\text{tl}_E s_0) \circ (\text{tl}_E s_1)))) \]

*Proof.* Let \( E \) be the context

\[
g : \Box(\{\text{Str}_E A \mid \Box[\varphi_0]\} \to \{\text{Str}_E A \mid \Box[\varphi_1]\}) \rightarrow \{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\},
\]

\[
s_0 : \{\text{Str}_E A \mid \Box[\varphi_0]\},
\]

\[
s_1 : \{\text{Str}_E A \mid \Box[\varphi_1]\}
\]

We have

\[
E \vdash \text{hd}_E s_0 : \{A \mid \varphi_0\} \quad E \vdash \text{tl}_E s_0 : \Box[\varphi_0] \]

\[
E \vdash \text{hd}_E s_1 : \{A \mid \varphi_1\} \quad E \vdash \text{tl}_E s_1 : \Box[\varphi_1]
\]

We thus get

\[
g \circ (\text{tl}_E s_0) \circ (\text{tl}_E s_1) : \Box(\{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\})
\]

and we are done since using subtyping we have

\[
\begin{align*}
\text{Cons}_E : \{A \mid \varphi_0\} & \rightarrow \Box(\{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\}) \rightarrow \{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\} \\
\text{Cons}_E : \{A \mid \varphi_1\} & \rightarrow \Box(\{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\}) \rightarrow \{\text{Str}_E A \mid \Box(\varphi_0 \lor \varphi_1)\}
\end{align*}
\]

\[ \square \]
E.2 Operations on Coinductive Streams

Example E.6 (Operations on Coinductive Streams). For a safe $\varphi$ of the appropriate type, we have

$$\begin{align*}
hd : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} & \rightarrow \{ A \mid \varphi \} \\
tl : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} & \rightarrow \{ \text{Str} A \mid \text{box} [\text{hd} \varphi] \} \\
tl : \{ \text{Str} A \mid \text{box} \odot \varphi \} & \rightarrow \{ \text{Str} A \mid \text{box} \varphi \}
\end{align*}$$

Proof.

Case of $\text{hd}$.

Recall that

$$\begin{align*}
\text{hd} : \text{Str} A & \rightarrow A \\
& := \lambda s. \text{hd}^\# \text{ (unbox } s) 
\end{align*}$$

We have

$$\begin{align*}
s : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} & \vdash s : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} \quad \square [\text{hd} \varphi] \text{ safe} \\
s & \vdash s : \square \{ \text{Str} A \mid \square [\text{hd} \varphi] \} \\
s & \vdash s : \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \vdash \text{unbox } s : \{ \text{Str} A \mid \square [\text{hd} \varphi] \} \\
s & \vdash s : \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \vdash \text{hd}^\# \text{ (unbox } s) : \{ A \mid \varphi \} \\
\vdash \lambda s. \text{hd}^\# \text{ (unbox } s) : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} & \rightarrow \{ A \mid \varphi \}
\end{align*}$$

Cases of $\text{tl}$.

Recall that

$$\begin{align*}
\text{tl} : \text{Str} A & \rightarrow \text{Str} A \\
& := \lambda s. \text{box}_\odot (\text{prev}_\odot (\text{tl}^\# \text{ (unbox } s))) 
\end{align*}$$

We have

$$\begin{align*}
s & : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} \vdash s : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} \\
s & \vdash s : \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \vdash \text{unbox } s : \square \{ \text{Str} A \mid \square [\text{hd} \varphi] \} \\
s & \vdash s : \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \vdash \text{tl}^\# \text{ (unbox } s) : \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \quad \text{Str } A \text{ constant} \\
s & \vdash s : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} \vdash \text{prev}_\odot (\text{tl}^\# \text{ (unbox } s)) : \{ \text{Str} A \mid \square [\text{hd} \varphi] \} \\
s & \vdash s : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} \vdash \text{box}_\odot (\text{prev}_\odot (\text{tl}^\# \text{ (unbox } s))) : \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \\
\vdash \lambda s. \text{box}_\odot (\text{prev}_\odot (\text{tl}^\# \text{ (unbox } s))) : \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \} & \rightarrow \{ \text{Str} A \mid \text{box} \square [\text{hd} \varphi] \}
\end{align*}$$

and

$$\begin{align*}
s & : \{ \text{Str} A \mid \text{box} \odot \varphi \} \vdash s : \{ \text{Str} A \mid \text{box} \odot \varphi \} \\
s & \vdash s : \text{Str} A \mid \text{box} \odot \varphi \vdash \text{unbox } s : \{ \text{Str} A \mid \odot \varphi \} \\
s & \vdash s : \text{Str} A \mid \text{box} \odot \varphi \vdash \text{tl}^\# \text{ (unbox } s) : \text{Str} A \mid \odot \varphi \quad \text{Str } A \text{ constant} \\
s & \vdash s : \{ \text{Str} A \mid \text{box} \odot \varphi \} \vdash \text{prev}_\odot (\text{tl}^\# \text{ (unbox } s)) : \{ \text{Str} A \mid \odot \varphi \} \\
s & \vdash s : \{ \text{Str} A \mid \text{box} \odot \varphi \} \vdash \text{box}_\odot (\text{prev}_\odot (\text{tl}^\# \text{ (unbox } s))) : \text{Str} A \mid \odot \varphi \\
\vdash \lambda s. \text{box}_\odot (\text{prev}_\odot (\text{tl}^\# \text{ (unbox } s))) : \{ \text{Str} A \mid \odot \varphi \} & \rightarrow \{ \text{Str } A \mid \odot \varphi \}
\end{align*}$$
E.3 Map over Coinductive Streams

We discuss here the cases of

\[\text{map} : \{B \mid \psi\} \rightarrow \{A \mid \varphi\} \rightarrow \{\text{Str } B \mid [\text{box}]\triangle[\text{hd}]\psi\} \rightarrow \{\text{Str } A \mid [\text{box}]\triangle[\text{hd}]\varphi\}\]

where \(\psi, \varphi\) are safe and smooth and where \(\triangle \in \{\Box, \Diamond, \Box\Diamond, \Diamond\Box\}\). The case of \(\Box\) is handled as in Ex. 5.4 using that \(\Box[\text{hd}]\varphi\) and \(\Box[\text{hd}]\psi\) are safe. The case of \(\Diamond\) is detailed in Ex. E.7 (E.3). The idea is that since \(\Diamond[\text{hd}]\varphi, \Diamond[\text{hd}]\psi\) are smooth and since \(\Diamond^k[\text{hd}]\varphi, \Diamond^k[\text{hd}]\psi\) are safe, we can reduce to typing the guarded map\(^E\) as

\[\text{map}^E : \{B \mid \psi\} \rightarrow \{A \mid \varphi\} \rightarrow \forall k \cdot \{\{\text{Str}^E B \mid \Diamond^k[\text{hd}]\psi\} \rightarrow \{\text{Str}^E A \mid \Diamond^k[\text{hd}]\varphi\}\}\]

The case of \(\Box\Diamond\), detailed in Ex. E.8 (E.3), is more involved. Since \(\Box\Diamond[\text{hd}]\varphi, \Diamond\Box[\text{hd}]\psi\) are smooth and \(\Diamond^k\Box[\text{hd}]\varphi, \Diamond^k\Box[\text{hd}]\psi\) are safe, we similarly reduce to showing (map\(^E\) \(f\)) : \(\forall k \cdot T(k)\) where

\[T(k) := \{\text{Str}^E B \mid \Diamond^k\Box[\text{hd}]\psi\} \rightarrow \{\text{Str}^E A \mid \Diamond^k[\text{hd}]\varphi\}\]

and assuming \(f\) of type \(\{B \mid \psi\} \rightarrow \{A \mid \varphi\}\). But this is unfortunately too weak. Similarly as with \(\Diamond\), it is natural to first assume the type \(\forall k \cdot T(k)\) for the recursion variable \(g\) and then to apply the \((\forall\text{-CI})\) rule (Fig. 9) on \(\forall k \cdot T(k)\). In the case of \(T(k+1)\), we unfold

\[\Diamond^{k+1}\Box[\text{hd}]\psi \leftrightarrow \Box[\text{hd}]\psi \lor \Box\Diamond\Box[\text{hd}]\psi\]

and apply the \((\forall\text-E)\) rule (Fig. 8). But in the branch of \(\Box[\text{hd}]\psi\), giving \(g\) the type, say,

\[\{\text{Str}^E B \mid \Diamond^k\Box[\text{hd}]\psi\} \rightarrow \{\text{Str}^E A \mid \Diamond^k\Box[\text{hd}]\varphi\}\]

is not sufficient to derive

\[s : \{\text{Str}^E B \mid \Box[\text{hd}]\psi\} \vdash g \otimes (\text{tl}^E s) : \text{Str}^E A \mid \Box[\text{hd}]\varphi\}\]

The reason is that \([\text{next}]\) (and thus \(\Diamond\)) does not satisfy axiom (P) of Table 2 (see §7). The solution is to use the \([\text{ev}(-)]\)%-> modality to encode a kind of “intersection” on arrow types, and to type (map\(^E\) \(f\)) with

\[\forall k \cdot \{\text{Str}^E B \rightarrow \text{Str}^E A \mid (\Diamond^k\Box[\text{hd}]\psi \rightarrow \Diamond^k\Box[\text{hd}]\varphi) \land (\Box[\text{hd}]\psi \rightarrow \Box[\text{hd}]\varphi)\}\]

We finally turn to \(\Box\Diamond\). Using that \(\Box\Diamond[\text{hd}]\varphi\) and \(\Box\Diamond[\text{hd}]\psi\) are both smooth, we first unfold the \(\Box\)’s using the rules \((\nu\text{-I})\) (Fig. 11) and then \((\nu\text{-E})\) (Ex. 6.10), thus reducing to

\[\text{box}_{\text{str}}(\text{map}^E \(f\) \text{ (unbox } s)) : \{\text{Str } A \mid [\text{box}]\Box^k\Diamond[\text{hd}]\varphi\}\]

assuming \(f : \{B \mid \psi\} \rightarrow \{A \mid \varphi\}\) and \(s : \{\text{Str } B \mid [\text{box}]\Box^k\Diamond[\text{hd}]\psi\}\). Then, since \(\Diamond[\text{hd}]\varphi, \Diamond[\text{hd}]\psi\) are smooth, we can unfold the \(\Diamond\)’s using the rules \((\mu\text{-E})\) and \((\mu\text{-I})\) with the non-trivial smooth context

\[\gamma(\beta) := \Box^k\beta\]
Since the formulae $\Box f \Diamond k \psi$ and $\Box f \Diamond k \psi$ are safe, we can reduce to showing

\[
\lambda \cdot \forall k \cdot U(\ell, k) = \{ \text{Str}^k B \mid \Box f \Diamond k \psi \} \rightarrow \{ \text{Str}^k A \mid \Box f \Diamond k \psi \}
\]

assuming \( f : \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \} \) and \( g : \forall \ell \cdot \forall k \cdot U(\ell, k) \). We apply the ($\forall$-CI) rule on \( \forall \ell \cdot \forall k \cdot U(\ell, k) \). The case of \( \forall k \cdot U(0, k) \) is trivial since \( \Box 0 \psi \rightarrow \top \). We then apply the ($\forall$-CI) rule, this time on \( \forall k \cdot U(\ell + 1, k) \). The case of \( U(\ell + 1, 0) \) can be dealt with using the (ExF) rule. In the case of \( U(\ell + 1, k + 1) \), we conclude with a straightforward case analysis based on the unfoldings

\[
\Box f \Diamond k \psi \rightarrow \{ \text{Str}^k B \mid \Diamond k \psi \} \rightarrow \{ \text{Str}^k A \mid \Diamond k \psi \}
\]

**The Case of Eventually ($\Diamond [hd] \varphi$)**

**Example E.7.** We have the following, for safe and smooth \( \varphi \) and \( \psi \):

\[
\text{map} : \{ \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \} \} \rightarrow \{ \text{Str} B \mid [\exists] \Diamond [hd] \psi \} \rightarrow \{ \text{Str} A \mid [\exists] \Diamond [hd] \varphi \}
\]

\[
= \lambda f. \lambda s. \text{box}_s (\text{map}^f f (\text{unbox} s))
\]

**Proof.** Since \( \Diamond [hd] \psi \) and \( \Diamond [hd] \varphi \) are both smooth, we can first reduce to

\[
E_f, s : \{ \text{Str} B \mid [\exists] \Diamond [hd] \psi \} \vdash \text{box}_s (\text{map}^f f (\text{unbox} s)) : \{ \text{Str} A \mid [\exists] \Diamond [hd] \varphi \}
\]

where

\[
E_f := f : \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \}
\]

Since the formulae \( \Diamond [hd] \psi \) and \( \Diamond [hd] \varphi \) are safe, we are done if we show

\[
\text{map}^f : \{ \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \} \} \rightarrow \forall k \cdot \{ \text{Str}^k B \mid \Diamond [hd] \psi \} \rightarrow \{ \text{Str}^k A \mid \Diamond [hd] \varphi \}
\]

\[
= \lambda f. \text{fix}(g). \lambda s. f (\text{hd}(s)) \rightarrow f (\text{hd}(s))
\]

Let

\[
N := (f (\text{hd}(s))) \rightarrow (g \circ (\text{hd}(s))
\]

\[
M := \lambda s. N
\]

\[
T(k) := \{ \text{Str}^k B \mid \Diamond [hd] \psi \} \rightarrow \{ \text{Str}^k A \mid \Diamond [hd] \varphi \}
\]

\[
E := E_f, g : \forall k \cdot T(k)
\]

We show

\[
E \vdash M : \forall k \cdot T(k)
\]

We reason by cases on \( k \) with the rule

\[
\frac{E \vdash M : T(0) \quad E \vdash M : T(k+1)}{E \vdash M : \forall k \cdot T(k)}
\]
Case of \( T(0) \).
We show
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond^0[hd] \psi \} \vdash N : \{ \text{Str}^\# A \mid \Diamond^0[hd] \varphi \}
\]
Since \( \vdash \Diamond^0[\psi] \iff \bot \), we conclude with the (EXF) rule
\[
\begin{array}{c}
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond^0[hd] \psi \} \vdash s : \{ \text{Str}^\# B \mid \bot \} \\
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond^0[hd] \psi \} \vdash N : \{ \text{Str}^\# A \mid \Diamond^0[hd] \varphi \}
\end{array}
\]

\[\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond^0[hd] \psi \} \vdash N : \{ \text{Str}^\# A \mid \Diamond^0[hd] \varphi \}\]

Case of \( T(k+1) \).
We show
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond^{k+1}[hd] \psi \} \vdash N : \{ \text{Str}^\# A \mid \Diamond^{k+1}[hd] \varphi \}
\]
Using
\[
\vdash \Diamond^{k+1}[hd] \psi \iff ([hd] \psi \lor \Diamond^k[hd] \psi)
\]
we do a case analysis on the refinement type of \( s \).

(\textbf{Sub})Case of \( [hd] \psi \).
Since \( \vdash [hd] \varphi \Rightarrow \Diamond^{k+1}[hd] \varphi \), we reduce to showing
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid [hd] \psi \} \vdash N : \{ \text{Str}^\# A \mid [hd] \varphi \}
\]
By \([E,1]\) we have
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid [hd] \psi \} \vdash \text{hd}^\# s : \{ B \mid \psi \}
\]
But we are done since
\[
\text{Cons}^\# : \{ A \mid \varphi \} \rightarrow \rightarrow \text{Str}^\# A \rightarrow \{ \text{Str}^\# A \mid [hd] \varphi \}
\]

(\textbf{Sub})Case of \( \Diamond \Diamond^k[hd] \psi \).
Since \( \vdash \Diamond \Diamond^k[hd] \varphi \Rightarrow \Diamond^{k+1}[hd] \varphi \), we reduce to showing
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond \Diamond^k[hd] \psi \} \vdash N : \{ \text{Str}^\# A \mid \Diamond \Diamond^k[hd] \varphi \}
\]
By \([E,1]\) we have
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond \Diamond^k[hd] \psi \} \vdash \text{tl}^\# s : \{ \text{Str}^\# B \mid \Diamond^{k}[hd] \psi \}
\]
Since
\[
\mathcal{E} \vdash g : \forall k : \rightarrow \{ \{ \text{Str}^\# B \mid \Diamond^k[hd] \psi \} \rightarrow \{ \text{Str}^\# A \mid \Diamond^k[hd] \varphi \} \}
\]
we have
\[
\mathcal{E} \vdash g : \rightarrow \{ \{ \text{Str}^\# B \mid \Diamond^k[hd] \psi \} \rightarrow \{ \text{Str}^\# A \mid \Diamond^k[hd] \varphi \} \}
\]
Since moreover by \([E,1]\) we have
\[
\text{Cons}^\# : A \rightarrow \rightarrow \{ \text{Str}^\# A \mid \Diamond^{k}[hd] \varphi \} \rightarrow \{ \text{Str}^\# A \mid \Diamond \Diamond^k[hd] \varphi \}
\]
we deduce that
\[
\mathcal{E}, s : \{ \text{Str}^\# B \mid \Diamond \Diamond^k[hd] \psi \} \vdash N : \{ \text{Str}^\# B \mid \Diamond \Diamond^k[hd] \psi \}
\]
\(\square\)
The Case of Eventually Always (\(\Diamond \Box [\text{hd}] \varphi\))

Example E.8. We have the following, for safe and smooth \(\varphi\) and \(\psi\):

\[
\begin{align*}
\text{map} & : (\{B \mid \psi\} \rightarrow \{A \mid \varphi\}) \rightarrow \{\text{Str} B \mid [\text{box}] \Diamond \Box [\text{hd}] \psi\} \rightarrow \{\text{Str} A \mid [\text{box}] \Diamond \Box [\text{hd}] \varphi\} \\
& = \lambda f. \lambda s. \text{box}_\Box (\text{map}^E f (\text{unbox} s))
\end{align*}
\]

Proof. Since \(\Diamond \Box [\text{hd}] \varphi\) and \(\Diamond \Box [\text{hd}] \psi\) are both smooth, we can first reduce to

\[
\begin{align*}
\mathcal{E}_f, \ s : \{\text{Str} B \mid [\text{box}] \Diamond \Box [\text{hd}] \psi\} & \vdash \text{box}_\Box (\text{map}^E f (\text{unbox} s)) : \{\text{Str} A \mid [\text{box}] \Diamond \Box [\text{hd}] \varphi\}
\end{align*}
\]

where

\[
\mathcal{E}_f := f : \{B \mid \psi\} \rightarrow \{A \mid \varphi\}
\]

Since the formulae \(\Diamond \Box [\text{hd}] \psi\) and \(\Diamond \Box [\text{hd}] \varphi\) are safe, we are done if we show

\[
\begin{align*}
\text{map}^E & : (\{B \mid \psi\} \rightarrow \{A \mid \varphi\}) \rightarrow \forall k : \{\text{Str}^E B \mid \Diamond \Box [\text{hd}] \psi\} \rightarrow \{\text{Str}^E A \mid \Diamond \Box [\text{hd}] \varphi\} \\
& = \lambda f. \lambda s. (f(\text{hd}^E s)) \vdash (g \circ (\text{tr}^E s))
\end{align*}
\]

Let

\[
\begin{align*}
N & := (f(\text{hd}^E s)) \vdash (g \circ (\text{tr}^E s)) \\
M & := \lambda s. N \\
T(k) & := \{\text{Str}^E B \rightarrow \text{Str}^E A \mid (\Diamond \Box [\text{hd}] \psi \rightarrow \Diamond \Box [\text{hd}] \varphi) \land (\Box [\text{hd}] \psi \rightarrow \Box [\text{hd}] \varphi)\} \\
\mathcal{E} & := \mathcal{E}_f, \ g : \forall k \cdot T(k)
\end{align*}
\]

We show

\[
\mathcal{E} \vdash M : \forall k \cdot T(k)
\]

We reason by cases on \(k\) with the rule

\[
\begin{array}{c}
\mathcal{E} \vdash M : T(0) \\
\mathcal{E} \vdash M : T(k+1)
\end{array}
\]

\[
\hline
\mathcal{E} \vdash M : \forall k \cdot T(k)
\]

Case of \(T(0)\).

We have to show

\[
\begin{align*}
\mathcal{E}, \ s : \{\text{Str}^E B \mid \Box [\text{hd}] \psi\} & \vdash N : \{\text{Str}^E A \mid \Box [\text{hd}] \varphi\} \\
\text{and} \quad \mathcal{E}, \ s : \{\text{Str}^E B \mid \Diamond \Box [\text{hd}] \psi\} & \vdash N : \{\text{Str}^E A \mid \Diamond \Box [\text{hd}] \varphi\}
\end{align*}
\]

We only detail the latter since the former can be dealt-with as in \(\|E, 1\|\). Since

\[
\vdash \Diamond \Box [\psi] \leftrightarrow \bot
\]

we conclude with the \((\text{ExF})\) rule

\[
\begin{align*}
\mathcal{E}, \ s : \{\text{Str}^E B \mid \Diamond \Box [\text{hd}] \psi\} & \vdash \bot \quad \mathcal{E}, \ s : \{\text{Str}^E B \mid \bot\} \\
\mathcal{E}, \ s : \{\text{Str}^E B \mid \Diamond \Box [\text{hd}] \psi\} & \vdash N : \text{Str}^E A \\
\mathcal{E}, \ s : \{\text{Str}^E B \mid \Diamond \Box [\text{hd}] \psi\} & \vdash N : \text{Str}^E A
\end{align*}
\]
Case of $T(k+1)$.

We show

$$\mathcal{E}, s : \{ \text{Str}^\delta B \mid \Box[\text{hd}]\varphi \} \vdash N : \{ \text{Str}^\delta A \mid \Box[\text{hd}]\varphi \}$$

and

$$\mathcal{E}, s : \{ \text{Str}^\delta B \mid (\Box^k + 1)[\text{hd}]\psi \} \vdash N : \{ \text{Str}^\delta A \mid \Diamond^k + 1[\text{hd}]\varphi \}$$

We only detail the latter since the former can be dealt-with as in §E.1. Using

$$\vdash \Diamond^k + 1[\text{hd}]\psi \iff (\Box[\text{hd}]\psi \lor \Box^k \Box[\text{hd}]\psi)$$

we do a case analysis on the refinement type of $s$.

(Sub)Case of $\Box[\text{hd}]\psi$.

We show

$$\mathcal{E}, s : \{ \text{Str}^\delta B \mid \Box[\text{hd}]\psi \} \vdash N : \{ \text{Str}^\delta A \mid \Box[\text{hd}]\varphi \}$$

and we can conclude as in §E.1.

(Sub)Case of $\Diamond^k[\text{hd}]\psi$.

Since $\vdash \Diamond^k[\text{hd}]\varphi \Rightarrow \Diamond^k[\text{hd}]\varphi$. We can therefore reduce to

$$\mathcal{E}, s : \{ \text{Str}^\delta B \mid \Diamond[\text{hd}]\psi \} \vdash N : \{ \text{Str}^\delta A \mid \Box[\text{hd}]\varphi \}$$

and we can conclude as in §E.1.

By §E.1 we have

$$\mathcal{E}, s : \{ \text{Str}^\delta B \mid \Box[\text{hd}]\psi \} \vdash N : \{ \text{Str}^\delta A \mid \Box[\text{hd}]\varphi \}$$

Since

$$\mathcal{E} \vdash g : \forall k \cdot \Box \left( \{ \text{Str}^\delta B \mid \Box^k [\text{hd}]\psi \} \rightarrow \{ \text{Str}^\delta A \mid \Box^k [\text{hd}]\varphi \} \right)$$

we have

$$\mathcal{E} \vdash g : \Box \left( \{ \text{Str}^\delta B \mid \Diamond[\text{hd}]\psi \} \rightarrow \{ \text{Str}^\delta A \mid \Box[\text{hd}]\varphi \} \right)$$

Since moreover by §E.1 we have

$$\text{Cons}^\delta : A \rightarrow \Box \{ \text{Str}^\delta A \mid \Diamond[\text{hd}]\varphi \} \rightarrow \{ \text{Str}^\delta A \mid \Box^k [\text{hd}]\varphi \}$$

we deduce that

$$\mathcal{E}, s : \{ \text{Str}^\delta B \mid \Box^k [\text{hd}]\psi \} \vdash N : \{ \text{Str}^\delta B \mid \Box^k [\text{hd}]\psi \}$$

$\Box$
The Case of Always Eventually \((\Box \Diamond [\text{hd}] \varphi)\)

**Example E.9.** We have the following, for safe and smooth \(\varphi\) and \(\psi\):

\[
\text{map} : \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \} \rightarrow \{ \text{Str} B \mid [\text{box}] \Box \Diamond [\text{hd}] \psi \} \rightarrow \{ \text{Str} A \mid [\text{box}] \Box \Diamond [\text{hd}] \varphi \} := \lambda f. \lambda s. \text{box}_{\alpha}(\text{map}^\chi f (\text{unbox} s))
\]

**Note E.10.** We let

\[
\begin{align*}
\Diamond^\chi \varphi &:= \mu^\chi \alpha. \varphi \lor \Box \alpha \\
\Box^\chi \varphi &:= \nu^\chi \alpha. \varphi \land \Box \alpha
\end{align*}
\]

**Proof.** We start in the same spirit as in §E.3 and §E.3. Using that \(\Box \Diamond [\text{hd}] \varphi\) and \(\Box \Diamond [\text{hd}] \psi\) are both smooth, we first unfold the \(\Box\) using the rules \((\nu - I)\) and \((\nu - E)\). Then, since \(\Box^\chi [\text{hd}] \varphi\) and \(\Box^\chi [\text{hd}] \psi\) are both smooth, we can unfold the \(\Diamond\) using the rules \((\mu - E)\) and \((\mu - I)\) with the non-trivial smooth context

\[
\gamma(\beta) := \Box^\chi \beta
\]

We are thus led to deriving

\[
\mathcal{E}_f, s : \{ \text{Str} B \mid [\text{box}] \Box^\chi \Diamond^\chi [\text{hd}] \psi \} \vdash \text{box}_{\alpha}(\text{map}^\chi f (\text{unbox} s)) : \{ \text{Str} A \mid [\text{box}] \Box^\chi \Diamond^\chi [\text{hd}] \varphi \}
\]

where

\[
\mathcal{E}_f := f : \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \}
\]

Since the formulae \(\Box^\chi \Diamond^\chi [\text{hd}] \psi\) and \(\Box^\chi \Diamond^\chi [\text{hd}] \varphi\) are safe, we are done if we show

\[
\text{map}^\chi : \{ B \mid \psi \} \rightarrow \{ A \mid \varphi \} \rightarrow \forall k \cdot \forall \ell \cdot \{ \text{Str}^\chi B \mid \Box^\chi \Diamond^\chi [\text{hd}] \psi \} \rightarrow \{ \text{Str}^\chi A \mid \Box^\chi \Diamond^\chi [\text{hd}] \varphi \}
\]

Let

\[
\begin{align*}
N &:= (f(\text{hd}^\chi s)) :\chi (g \circ (\text{tl}^\chi s)) \\
M &:= \lambda s. N \\
T(k, \ell) &:= \{ \text{Str}^\chi B \mid \Box^\chi \Diamond^\chi [\text{hd}] \psi \} \rightarrow \{ \text{Str}^\chi A \mid \Box^\chi \Diamond^\chi [\text{hd}] \varphi \}
\end{align*}
\]

We show

\[
\mathcal{E} \vdash M : \forall k \cdot \forall \ell \cdot T(k, \ell)
\]

We reason by cases on \(k\) and \(\ell\). This amounts to the derived rule

\[
\frac{
\mathcal{E} \vdash M : T(0, 0) \quad \mathcal{E} \vdash M : T(0, \ell + 1) \quad \mathcal{E} \vdash M : T(k + 1, 0) \quad \mathcal{E} \vdash M : T(k + 1, \ell + 1)
}{\mathcal{E} \vdash M : \forall k \cdot \forall \ell \cdot T(k, \ell)}
\]

**Cases of** \(T(u, 0)\).

We have \(\vdash \Box^\chi \theta \iff \top\), and we are done since

\[
\mathcal{E}, s : \{ \text{Str}^\chi B \mid \top \} \vdash N : \{ \text{Str}^\chi A \mid \top \}
\]
Case of $T(0, \ell+1)$.
We have $\vdash \Diamond^0[\theta] \Rightarrow \bot$, and we reduce to showing
\[
E, \ s : \{\text{Str}^E B \mid \Box^{\ell+1} \bot\} \vdash N : \{\text{Str}^E A \mid \Box^{\ell+1} \bot\}
\]
But since $\vdash \Box^{\ell+1} \Rightarrow \bot$, we have
\[
E, \ s : \{\text{Str}^E B \mid \Box^{\ell+1} \bot\} \vdash s : \{\text{Str}^E B \mid \bot\}
\]
and we conclude with the (ExF) rule
\[
\frac{\begin{array}{c}
E, \ s : \{\text{Str}^E B \mid \Box^{\ell+1} \bot\} \\
E, \ s : \{\text{Str}^E B \mid \Box^{\ell+1} \bot\} \vdash N : \text{Str}^E A
\end{array}}{E, \ s : \{\text{Str}^E B \mid \Box^{\ell+1} \bot\} \vdash N : \{\text{Str}^E A \mid \Box^{\ell+1} \bot\}}
\]

Case of $T(k+1, \ell+1)$.
Using $\vdash \text{Str}^A \Box^{\ell+1} \theta \iff (\theta \land \Diamond \Box^{\ell} \theta)$, we show
\[
E, \ s : \{\text{Str}^E B \mid \Box^{\ell+1} \Diamond^{k+1}[\text{hd}] \psi\} \vdash N : \{\text{Str}^E A \mid \Diamond^{k+1}[\text{hd}] \varphi \land \Diamond \Box^{\ell} \Diamond^{k+1}[\text{hd}] \varphi\}
\]
We consider each conjunct separately.

(Sub)Case of $\Diamond^{k+1}[\text{hd}] \varphi$.
We show
\[
E, \ s : \{\text{Str}^E B \mid \Diamond^{k+1}[\text{hd}] \psi\} \vdash N : \{\text{Str}^E A \mid \Diamond^{k+1}[\text{hd}] \varphi\}
\]
Using
\[
E, \ s : \{\text{Str}^E B \mid \Diamond^{k+1}[\text{hd}] \psi\} \vdash s : \{\text{Str}^E B \mid \Diamond^{k+1}[\text{hd}] \psi\}
\]
and $\vdash \Diamond^{k+1}[\text{hd}] \psi \iff ([\text{hd}] \psi \lor \Diamond \Diamond^{k}[\text{hd}] \psi)$ we do a case analysis on the refinement type of $s$.

(SubSub)Case of $[\text{hd}] \psi$.
Since (by [E.1])
\[
E, \ s : \{\text{Str}^E B \mid [\text{hd}] \psi\} \vdash \text{hd}\ s : \{\text{Str}^E B \mid [\text{hd}] \psi\}
\]
we easily deduce that
\[
E, \ s : \{\text{Str}^E B \mid [\text{hd}] \psi\} \vdash N : \{\text{Str}^E A \mid [\text{hd}] \varphi\}
\]
and we are done since $\vdash [\text{hd}] \varphi \Rightarrow \Diamond^{k+1}[\text{hd}] \varphi$.

(SubSub)Case of $\Diamond \Diamond^{k}[\text{hd}] \psi$.
By [E.1] we have
\[
E, \ s : \{\text{Str}^E B \mid \Diamond \Diamond^{k}[\text{hd}] \psi\} \vdash \text{tl}\ s : \{\text{Str}^E B \mid \Diamond^{k}[\text{hd}] \psi\}
\]
Since
\[
E \vdash g : \forall k \cdot \forall \ell \cdot \big( \{\text{Str}^E B \mid \Diamond \Diamond^{k}[\text{hd}] \psi\} \Rightarrow \{\text{Str}^E A \mid \Diamond \Diamond^{k}[\text{hd}] \varphi\} \big)
\]
we have

\[ E \vdash g : \{ \text{Str} \ B \mid \square^k \psi \} \rightarrow \{ \text{Str} \ A \mid \diamond^k \varphi \} \]

But \( \vdash (\theta \land \top) \Rightarrow \theta \), so that \( \vdash \square^1 \theta \Rightarrow \theta \), and thus

\[ E \vdash g : \{ \text{Str} \ B \mid \diamond^k \psi \} \rightarrow \{ \text{Str} \ A \mid \diamond^k \varphi \} \]

Since moreover by \( \{ E, I \} \) we have

\[ \text{Cons}^\xi : A \rightarrow \triangleright \{ \text{Str} \ A \mid \diamond^k \varphi \} \rightarrow \{ \text{Str} \ A \mid \lozenge^k \varphi \} \]

we deduce that

\[ E, s : \{ \text{Str} \ B \mid \lozenge^k \varphi \} \vdash N : \{ \text{Str} \ B \mid \lozenge^k \varphi \} \]

and we are done since \( \vdash \lozenge^k \varphi \Rightarrow \lozenge^{k+1} \varphi \).

(Sub)Case of \( \lozenge^f \lozenge^{k+1} \varphi \).

We show

\[ E, s : \{ \text{Str} \ B \mid \square^{f+1} \lozenge^{k+1} \varphi \} \vdash N : \{ \text{Str} \ B \mid \lozenge^k \varphi \} \]

by \( \{ E, I \} \) we have

\[ E, s : \{ \text{Str} \ B \mid \square^{f+1} \lozenge^{k+1} \varphi \} \vdash \text{tl}^\xi s : \triangleright \{ \text{Str} \ B \mid \lozenge^k \varphi \} \]

But now since

\[ E \vdash g : \forall k \cdot \forall \ell \cdot \triangleright \{ \text{Str} \ B \mid \square^f \lozenge^k \varphi \} \rightarrow \{ \text{Str} \ A \mid \lozenge^k \varphi \} \]

we have

\[ E \vdash g : \triangleright \{ \text{Str} \ B \mid \square^f \lozenge^k \varphi \} \rightarrow \{ \text{Str} \ A \mid \lozenge^k \varphi \} \]

and we conclude with \( \{ E, I \} \) namely

\[ \text{Cons}^\xi : A \rightarrow \triangleright \{ \text{Str} \ A \mid \square^f \lozenge^{k+1} \varphi \} \rightarrow \{ \text{Str} \ A \mid \lozenge^f \lozenge^{k+1} \varphi \} \]

\( \square \)

E.4 The Diagonal Function

Consider a stream of streams \( s \). We have \( s = (s_i \mid i \geq 0) \) where each \( s_i \) is itself a stream \( s_i = (s_{i,j} \mid j \geq 0) \). The diagonal of \( s \) is then the stream \( (s_{i,i} \mid i \geq 0) \). Note that \( s_{i,i} = \text{hd}(\text{tl}^\xi(\text{hd}(\text{tl}^\xi(s)))) \). Indeed, \( \text{tl}^\xi(s) \) is the stream of streams \( (s_k \mid k \geq i) \),
Recall that for a

The Guarded Diagonal Function

Similarly as for map we obtain

In Ex. E.12 (§ diagaux

This easily follows from the fact that using Ex. 5.3 and Ex. 5.4, we can type

The auxiliary higher-order function diagaux iterates the coinductive tl over the head of the stream of streams s. We write o for function composition, so that assuming s : Str^k(Str A) and t : Str A → Str A, we have

This requires the coinductive type Str A. In Ex. E.11 (§ E.4) below, for a safe \( \varphi \) we obtain

This easily follows from the fact that using Ex. 5.3 and Ex. 5.4 we can type diagaux with

In Ex. E.12 (§ E.4) we show that for a safe and smooth \( \varphi \), we have

Similarly as for map in [E.3] we reduce to

The Guarded Diagonal Function

Example E.11 (The Guarded Diagonal Function). For a safe \( \varphi \), we have

Recall that

so that \( \text{hd}(\text{tl}^i(s)) \) is the stream \( s_i \) and \( \text{tl}^i(\text{hd}(\text{tl}^i(s))) \) is the stream \( (s_{i,k} | k \geq i) \). Taking its the head thus gives \( s_{i,k} \).

We implement the diagonal function as follows:

\[
\text{diag} := \lambda s. \text{box}((\text{diag}^\varphi(\text{unbox } s)) : \text{Str}(\text{Str } A) \rightarrow \text{Str } A
\]

\[
\text{diag}^\varphi := \text{diagaux}^\varphi \text{id} : \text{Str}^\varphi(\text{Str } A) \rightarrow \text{Str}^\varphi A
\]

\[
\text{diagaux}^\varphi : (\text{Str } A \rightarrow \text{Str } A) \rightarrow \text{Str}^\varphi(\text{Str } A) \rightarrow \text{Str}^\varphi A
\]

\[
:= \text{fix}(g). \forall. \text{As}. \text{Cons}^\varphi ((\text{hd } o t)(\text{hd}^\varphi s)) \ (g \ominus \text{next}(t o t) \ominus (\text{tl}^\varphi s))
\]
Proof. We reduce to
\[
\text{diagaux}^\varphi : \ (\{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \}) \to \ (\text{Str}^\varphi(\text{Str} A) \mid \square \text{hd} \top \text{box} \square \text{hd} \varphi) \to \{ \text{Str} A \mid \square \text{hd} \varphi \}
\]

Let \( \mathcal{E} \) be the context
\[
g : \top, \\
t : \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \}, \\
s : \text{Str}^\varphi(\text{Str} A) \mid \square \text{hd} \top \text{box} \square \text{hd} \varphi
\]
where \( T \) is the type
\[
(\{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \}) \to \ (\text{Str}^\varphi(\text{Str} A) \mid \square \text{hd} \top \text{box} \square \text{hd} \varphi) \to \{ \text{Str} A \mid \square \text{hd} \varphi \}
\]
The result directly follows from the following typings, which are themselves given by \[\ref{E.1} \] \[\ref{E.1} \] and \[\ref{E.2} \]
\[
\mathcal{E} \vdash \text{hd} \circ t : \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \{ A \mid \varphi \}, \\
\mathcal{E} \vdash \text{hd}^\varphi \circ s : \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \}, \\
\mathcal{E} \vdash t \circ tl : \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \}, \\
\mathcal{E} \vdash tl^\varphi s : \{ \text{Str}^\varphi(\text{Str} A) \mid \square \text{hd} \top \text{box} \square \text{hd} \varphi \}
\]

\( \square \)

The Coinductive Diagonal Function

Example E.12 (The Coinductive Diagonal Function). For a safe and smooth \( \varphi \), we have
\[
\text{diag} : \{ \text{Str}(\text{Str} A) \mid \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \}
\]
\[
:= \lambda s. \text{box}(\text{diag}^\varphi(\text{unbox} s))
\]

Proof. Using that \( \square \varphi \top \text{hd} \top \text{box} \square \text{hd} \varphi \) and \( \square \varphi \top \text{hd} \varphi \) are both smooth, we can first reduce to
\[
s : \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \text{box}_i(\text{diag}^\varphi(\text{unbox} s)) : \{ \text{Str} A \mid \text{box} \square \varphi \top \text{hd} \varphi \}
\]
Since the formulae \( \square \varphi \top \text{hd} \top \text{box} \square \text{hd} \varphi \) and \( \square \varphi \top \text{hd} \varphi \) are safe, we are done if we show
\[
\text{diag}^\varphi : \forall k : (\{ \text{Str}^\varphi(\text{Str} A) \mid \square \varphi \top \text{hd} \top \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \square \varphi \top \text{hd} \varphi \})
\]
Consider the types
\[
U(k) := \{ \text{Str}^\varphi(\text{Str} A) \to \text{Str}^\varphi A \mid \psi_0 \land \psi_1 \} \\
T(k) := \{ \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \to \{ \text{Str} A \mid \text{box} \square \text{hd} \varphi \} \} \to U(k)
\]
where
\[ \psi_0 := \Diamond^k \Box \llhd [\Box [\Box [\text{hd}] \varphi] |\rightarrow \Diamond^k \Box [\Box [\text{hd}] \varphi] \]
\[ \psi_1 := \Box [\text{hd}] [\Box [\Box [\text{hd}] \varphi] |\rightarrow \Box [\text{hd}] \varphi \]

We show
\[ \text{dia} \gamma : \forall k \cdot T(k) \]

Let
\[ N := \text{Cons} \gamma ((\text{hd} \circ t)(\text{hd} \gamma s)) \]
\[ M := \lambda g. \lambda s. N \]
\[ \mathcal{E} := g : \Box \forall k \cdot T(k) \]

We reason by cases on \( k \) with the rule
\[
\frac{\mathcal{E} \vdash M : T(0) \quad \mathcal{E} \vdash M : T(k+1)}{\mathcal{E} \vdash M : \forall k \cdot T(k)}
\]

Let
\[ \mathcal{E}' := \mathcal{E}, \; t : \{ \text{Str } A \mid [\Box [\Box [\text{hd}] \varphi] \} \rightarrow \{ \text{Str } A \mid [\Box [\Box [\text{hd}] \varphi] \}
\]
We omit the proof of
\[ \mathcal{E}' \vdash \lambda s. N : \{ \text{Str}^\gamma (\text{Str } A) \rightarrow \text{Str}^\gamma A \mid [\text{ev} (\Box [\Box [\text{hd}] \varphi])] [\Box [\Box [\text{hd}] \varphi] \}
\]
since it follows that of \([E3]\).

**Case of \( T(0) \).
**
Since \( \vdash \Diamond^0 \theta \leftrightarrow \bot \), we reduce to showing
\[ \mathcal{E} \vdash \lambda t. \lambda s. N : (\{ \text{Str } A \mid [\Box [\Box [\text{hd}] \varphi] \rightarrow \{ \text{Str } A \mid [\Box [\Box [\text{hd}] \varphi] \}) \rightarrow \{ \text{Str}^\gamma (\text{Str } A) \mid \bot \} \rightarrow \{ \text{Str}^\gamma A \mid \Diamond^0 [\Box [\Box [\text{hd}] \varphi] \}
\]
and we conclude using the (EXF) rule.

**Case of \( T(k+1) \).
**
We show
\[ \mathcal{E}', \; s : \{ \text{Str}^\gamma (\text{Str } A) \mid \Diamond^{k+1} [\Box [\Box [\text{hd}] \varphi] \rightarrow \{ \text{Str}^\gamma A \mid \Diamond^{k+1} [\Box [\Box [\text{hd}] \varphi] \}
\]
Using
\[ \vdash \Diamond^{k+1} \theta \iff \theta \lor \Diamond \Diamond \theta \]
we reason by cases on the refinement of \( s \). This leads to two subcases.

**Subcase** of \( [\Box [\Box [\text{hd}] \varphi] \Box [\Box [\text{hd}] \varphi].
**
We show
\[ \mathcal{E}', \; s : \{ \text{Str}^\gamma (\text{Str } A) \mid [\Box [\Box [\text{hd}] \varphi] \rightarrow \{ \text{Str}^\gamma A \mid \Diamond^{k+1} [\Box [\Box [\text{hd}] \varphi] \}
\]
Since \( \vdash [\Box [\Box [\text{hd}] \varphi] \rightarrow \Diamond^{k+1} [\Box [\Box [\text{hd}] \varphi] \), we can reduce to
\[ \mathcal{E}', \; s : \{ \text{Str}^\gamma (\text{Str } A) \mid [\Box [\Box [\text{hd}] \varphi] \rightarrow \{ \text{Str}^\gamma A \mid [\Box [\Box [\text{hd}] \varphi] \}
\]
which is proved as in \([E3]\).
(Sub)Case of \( \varnothing \diamond k \diamond [\text{hd}][\text{box}][\text{hd}] \varphi \).
We show that \( \mathcal{E}', s : \{ \text{Str}^k(\text{Str} A) \mid \varnothing \diamond k \diamond [\text{hd}][\text{box}][\text{hd}] \varphi \} \vdash N : \{ \text{Str}^k A \mid \varnothing \diamond k \diamond [\text{hd}] \varphi \} \)

Let \( \mathcal{E}'' := \mathcal{E}', s : \{ \text{Str}^k(\text{Str} A) \mid \varnothing \diamond k \diamond [\text{hd}][\text{box}][\text{hd}] \varphi \} \)

Note that \( \mathcal{E}'' \vdash g : \mathsf{T}(k) \), so that by Ex. 2 we have

\[ \mathcal{E}'' \vdash g \circ \text{next}(t \circ tl) : \mathsf{T}(\{ \text{Str}^k(\text{Str} A) \mid \varnothing \diamond k \diamond [\text{hd}][\text{box}][\text{hd}] \varphi \} \rightarrow \{ \text{Str}^k A \mid \varnothing \diamond k \diamond [\text{hd}] \varphi \} \)

Using Ex. 1 we derive

\[ \mathcal{E}'' \vdash s : \{ \text{Str}^k(\text{Str} A) \mid \varnothing \diamond k \diamond [\text{hd}][\text{box}][\text{hd}] \varphi \} \]

\[ \mathcal{E}'' \vdash t^k s : \mathsf{T}(\{ \text{Str}^k(\text{Str} A) \mid \varnothing \diamond k \diamond [\text{hd}][\text{box}][\text{hd}] \varphi \} \rightarrow \{ \text{Str}^k A \mid \varnothing \diamond k \diamond [\text{hd}] \varphi \}) \]

\[ \mathcal{E}'' \vdash g \circ \text{next}(t \circ tl) \circ (t^k s) : \mathsf{T}(\{ \text{Str}^k A \mid \varnothing \diamond k \diamond [\text{hd}] \varphi \}) \]

\[ \mathcal{E}'' \vdash \text{Cons}^k ((\text{hd} \circ t)(\text{hd}^k s)) (g \circ \text{next}(t \circ tl) \circ (t^k s)) : \{ \text{Str}^k A \mid \varnothing \diamond k \diamond [\text{hd}] \varphi \} \]

\[ \square \]

### E.5 Fair Streams

We discuss here an adaptation of the *fair streams* of [18, 8]. We rely on the basic datatypes presented in E.5. In E.5 we discuss a function

\[ \text{fb} : \text{CoNat} \rightarrow \text{CoNat} \rightarrow \text{Str} \text{Bool} \]

such that, writing 0 for Z and 1 for (S Z) (see Ex. E.15), the *non-regular* stream (fb 0 1), adapted from [18, 8], is of the form

\[ \text{ff tt ff tt tt ff tt tt tt ff tt tt tt ff tt tt tt ff ff ...} \]

This stream thus contains infinitely many tt’s and infinitely many ff’s. This is expressed with the formula \( [\text{box}]\varnothing \diamond [\text{hd}] [\text{tt}] \land [\text{box}]\varnothing \diamond [\text{hd}] [\text{ff}] \) where [tt], [ff] represent the value of a Boolean, as in

\[ \text{tt} : \{ \text{Bool} \mid [\text{tt}] \} \] and \( \text{ff} : \{ \text{Bool} \mid [\text{ff}] \} \)

Examples E.20 and E.22 show that we indeed have

\[ (\text{fb 0 1}) : \{ \text{Str} \text{Bool} \mid [\text{box}]\varnothing \diamond [\text{hd}][\text{tt}] \land [\text{box}]\varnothing \diamond [\text{hd}][\text{ff}] \} \]

The key are the following refinement typings for the *guarded* fb\(^k\), discussed in Ex. E.19 and Ex. E.21.

\[ \text{fb}^k : \text{CoNat}^k \rightarrow \{ \text{CoNat}^k \mid [S] \} \rightarrow \{ \text{Str}^k \text{Bool} \mid \varnothing (\text{hd})[\text{tt}] \lor \varnothing (\text{hd})[\text{tt}] \} \]

\[ \text{fb}^k : \forall k : \forall t : \{ \text{CoNat}^k \mid \varnothing^k [Z] \} \rightarrow \{ \text{CoNat}^k \mid \varnothing^{k+1} [Z] \} \rightarrow \{ \text{Str}^k \text{Bool} \mid \varnothing^k \varnothing^{k+1} [\text{hd}][\text{ff}] \} \]
where, as in Not. E.10 (§E.3), we let
\[ \Box^\nu \varphi := \nu^\alpha \varphi \land \Box^\alpha \]
Finally, in §E.5 we discuss a stream scheduler
\[ \text{sched} : \text{Str Bool} \rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \text{Str}(A + B) \]
such that \(\text{sched}\) can be typed as follows (Ex. E.25):
\[ \{\text{Str Bool} | [\text{box}]\Box \land [\text{hd}]tt\} \rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \{\text{Str}(A + B) | [\text{box}]\Box \land [\text{hd}]in_0\}T \]
\[ \{\text{Str Bool} | [\text{box}]\Box \land [\text{hd}]ff\} \rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \{\text{Str}(A + B) | [\text{box}]\Box \land [\text{hd}]in_1\}T \]
and thus
\[ \text{sched}(fb 0 1) : \{\text{Str}(A + B) | [\text{box}]\Box \land [\text{hd}]in_0\}T \land [\text{box}]\Box \land [\text{hd}]in_1\}T \}

**Basic Datatypes**

*Example E.13 (Booleans).* Let
\[ \text{Bool} := 1 + 1 \]
with constructors
\[ \text{tt} := \text{in}_0(()) : \text{Bool} \]
\[ \text{ff} := \text{in}_1(()) : \text{Bool} \]

*Example E.14 (Formulae on Booleans).*
\[ [\text{tt}] := [\text{in}_0]T : \text{Bool} \]
\[ [\text{ff}] := [\text{in}_1]T : \text{Bool} \]

*Example E.15 (CoNatural Numbers).* Let
\[ \text{CoNat} := \Box \text{CoNat}^g \]
\[ \text{CoNat}^g := \text{Fix}(X).1 + \Box X \]
with constructors
\[ Z := \text{box}_n(Z^g) : \text{CoNat} \]
\[ S := \lambda n.\text{box}_n(S^g (\text{unbox} n)) : \text{CoNat} \rightarrow \text{CoNat} \]
\[ Z^g := \text{fold}(\text{in}_0(()) : \text{CoNat}^g \]
\[ S^g := \lambda n.\text{fold}(\text{in}_1 n) : \Box \text{CoNat}^g \rightarrow \text{CoNat}^g \]

*Example E.16 (Formulae on CoNatural Numbers).*
\[ [Z] := [\text{fold}[\text{in}_0]] : \text{CoNat}^g \]
\[ [S] := [\text{fold}[\text{in}_1]] : \text{CoNat}^g \]
\[ \Box \varphi := [\text{fold}[\text{in}_1][\text{next}]\varphi : \text{CoNat}^g \]
\[ \Diamond \varphi := \mu \alpha. \varphi \lor \Box \alpha : \text{CoNat}^g \]
\[ \Diamond^\nu \varphi := \mu^\alpha \varphi \lor \Box^\alpha : \text{CoNat}^g \]
where \(\varphi : \text{CoNat}^g\).
A Fair Stream of Booleans

Example E.17.

\[ fb : \text{CoNat} \rightarrow \text{CoNat} \rightarrow \text{Str Bool} \]
\[ := \lambda c. \lambda m. \text{box}_c (fb^g (\text{unbox} \ c) (\text{unbox} \ m)) \]

\[ fb^g : \text{CoNat}^g \rightarrow \text{CoNat}^g \rightarrow \text{Str}^g \text{Bool} \]
\[ := \text{fix}(g). \lambda c. \lambda m. \text{case} \ c \ of \]
\[ | Z^g \mapsto ff^g \circ (\text{next} \ m) \circ \text{next}(S^g (\text{next} \ m)) \]
\[ | S^g n \mapsto \text{tt}^g \circ n \circ (\text{next} \ m) \]

Example E.18.

\[ fb : \{(\text{CoNat} | \{\text{box}\}Z)\} \rightarrow \text{CoNat} \rightarrow \{\text{Str Bool} | \{\text{box}\}hd[ff]\} \]
\[ fb^g : \forall k \cdot \{(\text{CoNat}^g | \diamond^k Z)\} \rightarrow \text{CoNat}^g \rightarrow \{\text{Str}^g \text{Bool} | \diamond^k [\text{hd}[ff]]\} \]

Proof. Let

\[ T(k) := \{(\text{CoNat}^g | \diamond^k Z)\} \rightarrow \text{CoNat}^g \rightarrow \{\text{Str}^g \text{Bool} | \diamond^k [\text{hd}[ff]]\} \]

and assume

\[ g : \vdash \forall k \cdot T(k) \]

Let

\[ M(g, c, m) := \text{case} \ c \ of \]
\[ | Z^g \mapsto ff^g \circ (\text{next} \ m) \circ \text{next}(S^g (\text{next} \ m)) \]
\[ | S^g n \mapsto \text{tt}^g \circ n \circ (\text{next} \ m) \]

We show

\[ \lambda c. \lambda m. M(g, c, m) : \forall k \cdot T(k) \]

We apply the (\forall -CI) rule on \forall k. This leads to two cases.

Case of \(T(0)\). We get the result from the (ExF) rule since

\[ \diamond^0 Z \Rightarrow \bot \]

Case of \(T(k+1)\). We show

\[ M(g, c, m) : \{\text{Str}^g \text{Bool} | \diamond^{k+1} [\text{hd}[ff]]\} \]

assuming

\[ c : \{(\text{CoNat}^g | \diamond^{k+1} Z)\} \]
\[ m : \text{CoNat}^g \]

Using

\[ \diamond^{k+1} Z \Rightarrow [Z] \lor \diamond^k Z \]

we reason by cases on the refinement type of \(c\). This leads to two subcases.
(Sub)Case of \([Z]\). We apply the \((\text{INJ}_0 - \text{E})\) rule on the refinement type of 
\((\text{unfold } c)\). Since 
\[ \text{hd}[\text{ff}] \Rightarrow \diamond^{k+1}[\text{hd}][\text{ff}] \]
the result follows from the fact that 
\[ \text{ff} :: g \odot (\text{next } m) \odot \text{next}(S(\text{next } m)) : \{\text{Str}^g \text{Bool} | \text{hd}[\text{ff}]\} \]

(Sub)Case of \(\odot \diamond^k[Z]\). We have 
\[ \text{unfold } c : \{1 + \triangleright \text{CoNat}^g | [\text{in}_1][\text{next}\diamond^k] Z\} \]
By applying the \((\text{INJ}_1 - \text{E})\) rule on the refinement type of \((\text{unfold } c)\), we are left with showing 
\[ \text{tt} :: g \odot n \odot (\text{next } m) : \{\text{Str}^g \text{Bool} | \diamond^{k+1}[\text{hd}][\text{ff}]\} \]
assuming 
\[ n : \triangleright \{\text{CoNat}^g | \diamond^k[Z]\} \]
Using 
\[ \odot \diamond^k[\text{hd}][\text{ff}] \Rightarrow \diamond^{k+1}[\text{hd}][\text{ff}] \]
we are done since 
\[ g \odot n \odot (\text{next } m) : \triangleright \{\text{Str}^g \text{Bool} | \diamond^k[\text{hd}][\text{ff}]\} \]
\[ \square \]

**Example E.19.** Consider a function 
\[ f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \]
such that 
\[- 1 \leq f(k + 1, \ell + 1) \]
\[- f(k, \ell + 2) \leq f(k + 1, \ell + 1) \]
\[- \ell + 1 \leq f(k + 1, \ell + 1) \]
\[- f(k, \ell + 1) \leq f(k + 1, \ell + 1) \]
for instance \(f(k, \ell) = k + \ell\). Then we can give the following refined type to \(\text{fb}^g\):
\[ \forall k \forall \ell. \left( \{\text{CoNat}^g | \diamond^k[Z]\} \rightarrow \{\text{CoNat}^g | \diamond^{k+1}[Z]\} \rightarrow \{\text{Str}^g \text{Bool} | \square^k \diamond^k(\ell)[\text{hd}][\text{ff}]\} \right) \]

**Proof.** Let 
\[ U(k, \ell) := \{\text{CoNat}^g \rightarrow \text{CoNat}^g \rightarrow \text{Str}^g \text{Bool} | \varphi(k, \ell) \land \psi(\ell)\} \]
\[ \varphi(k, \ell) := \diamond^k[Z] \Rightarrow \diamond^{k+1}[Z] \Rightarrow \square^k \diamond^k(\ell)[\text{hd}][\text{ff}] \]
\[ \psi(\ell) := \diamond^\ell[Z] \Rightarrow \top \Rightarrow \diamond^\ell[\text{hd}][\text{ff}] \]
and assume 
\[ g : \triangleright \forall k \cdot \forall \ell \cdot U(k) \]
Let
\[ M(g, c, m) := \begin{cases} Z \mapsto \text{ff} & \cdot g \odot \text{next m} \odot \text{next}(S^g \text{ (next m)}) \\ S^n \mapsto \text{tt} & \cdot g \odot n \odot \text{next m} \end{cases} \]

We show
\[ \lambda c. \lambda m. M(g, c, m) : \forall k \cdot \forall \ell \cdot U(k) \]

First, proceeding similarly as in Ex. E.18,
\[ \lambda c. \lambda m. M(g, c, m) : \forall \ell \cdot \{ \text{CoNat}^g \rightarrow \text{CoNat}^g \rightarrow \text{Str}^g \text{ Bool} \mid \Diamond^\ell[Z] \rightarrow \top \rightarrow \Diamond^\ell[\text{hd}[\text{ff}]] \} \]

Let
\[ T(k, \ell) := \{ \text{CoNat}^g \mid \Diamond^\ell[Z] \} \rightarrow \{ \text{CoNat}^g \mid \Diamond^{\ell+1}[Z] \} \rightarrow \{ \text{Str}^g \text{ Bool} \mid \Box^k \Diamond^{(k, \ell)}[\text{hd}][\text{ff}] \} \]

We show
\[ \lambda c. \lambda m. M(g, c, m) : \forall k \cdot \forall \ell \cdot T(k) \]

We apply the \((\forall\text{-CI})\) rule on \(\forall k\). In the case of \(\forall \ell \cdot T(0, \ell)\), the result is trivial since
\[ \Box^0 \Diamond^{(0, \ell)}[\text{hd}][\text{ff}] \leftrightarrow \top \]

In the case of \(\forall \ell \cdot T(k+1, \ell)\), we apply the \((\forall\text{-CI})\) rule, this time on \(\forall \ell\). The case of \(T(k+1, 0)\) is dealt-with using the \((\text{ExF})\) rule since
\[ \Diamond^0[Z] \leftrightarrow \bot \]

In the case of \(T(k+1, \ell+1)\), we show
\[ M(g, c, m) : \{ \text{Str}^g \text{ Bool} \mid \Box^{k+1} \Diamond^{(k+1, \ell+1)}[\text{hd}][\text{ff}] \} \]

assuming
\[ c : \{ \text{CoNat}^g \mid \Diamond^{\ell+1}[Z] \} \]
\[ m : \{ \text{CoNat}^g \mid \Diamond^{\ell+2}[Z] \} \]

We apply the typing rule for case (Fig. 4). This leads to two branches, one for \((\text{unfold } c) = \text{fold}(\text{in}_0())\) (denoted \(Z^g\)), and one for \((\text{unfold } c) = \text{fold}(\text{in}_1 n)\) (denoted \(S^n\)).

**Case of \(Z^g\).**

We have to show
\[ \text{ff} :: g \odot (\text{next m}) \odot \text{next}(S \text{ (next m)}) : \{ \text{Str}^g \text{ Bool} \mid \Box^{k+1} \Diamond^{(k+1, \ell+1)}[\text{hd}][\text{ff}] \} \]

We have
\[ \Box^{k+1} \Diamond^{(k+1, \ell+1)}[\text{hd}][\text{ff}] \leftrightarrow \Diamond^{(k+1, \ell+1)}[\text{hd}][\text{ff}] \land \Box^k \Diamond^{(k+1, \ell+1)}[\text{hd}][\text{ff}] \]

and we consider each conjunct separately.
(Sub)Case of $\diamondsuit f^{(k+1,ℓ+1)}[hd][ff]$. 
We have

$$ff ::\overset{\varepsilon}{g \oplus (next\ m) \oplus next(S\ (next\ m)) : \{Str^kBool \mid [hd][ff]}$$

and as $f(k+1,ℓ+1) \geq 1$ we are done with

$$[hd][ff] \Rightarrow \diamondsuit f^{(k+1,ℓ+1)}[hd][ff]$$

(Sub)Case of $\square k \diamondsuit f^{(k+1,ℓ+1)}[hd][ff]$. 
Since

$$m : \{CoNat^k \mid \diamondsuit f^{\ell+2}[Z]\}$$

$$S^k\ (next\ m) : \{CoNat^k \mid \diamondsuit f^{\ell+3}[Z]\}$$

we have

$$g \oplus (next\ m) \oplus next(S\ (next\ m)) : \rightarrow \{Str^kBool \mid \square k \diamondsuit f^{(k,ℓ+2)}[hd][ff]}$$

so that

$$ff ::\overset{\varepsilon}{g \oplus (next\ m) \oplus next(S\ (next\ m)) : \{Str^kBool \mid \square k \diamondsuit f^{(k,ℓ+2)}[hd][ff]}$$

But since $f(k,ℓ+2) \leq f(k+1,ℓ+1)$, we have

$$\diamondsuit f^{(k,ℓ+2)}[hd][ff] \Rightarrow \diamondsuit f^{(k+1,ℓ+1)}[hd][ff]$$

and we obtain

$$ff ::\overset{\varepsilon}{g \oplus (next\ m) \oplus next(S\ (next\ m)) : \{Str^kBool \mid \square k \diamondsuit f^{(k+1,ℓ+1)}[hd][ff]}$$

Case of $S^k n$. 
We have to show

$$tt ::\overset{\varepsilon}{g \oplus n \oplus (next\ m) : \{Str^kBool \mid \square k+1 \diamondsuit f^{(k+1,ℓ+1)}[hd][ff]}$$

assuming

$$n : \{CoNat^k \mid \diamondsuit [Z]\}$$

We have

$$\square k+1 \diamondsuit f^{(k+1,ℓ+1)}[hd][ff] \Rightarrow \diamondsuit f^{(k+1,ℓ+1)}[hd][ff] \land \square k \diamondsuit f^{(k+1,ℓ+1)}[hd][ff]$$

and we consider each conjunct separately.

(Sub)Case of $\diamondsuit f^{(k+1,ℓ+1)}[hd][ff]$. 
Using

$$g : \rightarrow \{CoNat^k \rightarrow CoNat^k \rightarrow Str^kBool \mid \diamondsuit [Z] \rightarrow \top \land \rightarrow \diamondsuit [hd][ff}\}$$

we get

$$tt ::\overset{\varepsilon}{g \oplus n \oplus (next\ m) : \{Str^kBool \mid \diamondsuit [hd][ff]}$$

and the result follows from the fact that

$$ℓ + 1 \leq f(k+1,ℓ+1)$$
(Sub)Case of $\bigcirc k \Diamond \ell [k+1, \ell+1]^{[hd][ff]}$.
Since $\ell \leq \ell + 1$, we have

$$n : \{\text{CoNat}^g \mid \Diamond^{\ell+1}[Z]\}$$

and thus

$$g \star n \star (\text{next } m) : \rightarrow \{\text{Str}^g \mid □k \Diamond^{(k, \ell+1)}[hd][ff]\}$$

so that

$$\text{tt} :\rightarrow g \star n \star (\text{next } m) : \{\text{Str}^g \mid □k \Diamond^{(k, \ell+1)}[hd][ff]\}$$

But since $f(k, \ell + 1) \leq f(k + 1, \ell + 1)$ we have

$$\Diamond^{(k, \ell+1)}[hd][ff] \Rightarrow \Diamond^{(k+1, \ell+1)}[hd][ff]$$

and we obtain

$$\text{tt} :\rightarrow g \star n \star (\text{next } m) : \{\text{Str}^g \mid □k \Diamond^{(k+1, \ell+1)}[hd][ff]\}$$

Example E.20. We have

$$\text{fb } Z \ (S \ Z) : \{\text{Str } \mid \boxed{[\square \Diamond]}[hd][ff]\}$$

Proof. Recall that

$$\text{fb} : \text{CoNat} \rightarrow \text{CoNat} \rightarrow \text{Str } \text{Bool}$$
$$:= \lambda c.\lambda m. \text{box}_c((\text{fb}^g \ (\text{unbox } c) \ (\text{unbox } m)))$$

We show

$$\text{fb} : \forall \ell : \{\{\text{CoNat} \mid \boxed{[\square \Diamond]}[Z]\} \rightarrow \{\text{CoNat} \mid \boxed{[\square \Diamond]}^{\ell+1}[Z]\} \rightarrow \{\text{Str } \mid \boxed{[\square \Diamond]}[hd][ff]\}$$

We apply the ($\forall$-I) rule. Assume

$$c : \{\text{CoNat} \mid \boxed{[\square \Diamond]}[Z]\}$$
$$m : \{\text{CoNat} \mid \boxed{[\square \Diamond]}^\ell [Z]\}$$

Since the formulae $\Diamond^{\ell} [Z]$ and $\Diamond^{\ell+1} [Z]$ are safe we have

$$c : \square \{\text{CoNat}^g \mid \Diamond^{\ell} [Z]\}$$
$$m : \square \{\text{CoNat}^g \mid \Diamond^{\ell+1} [Z]\}$$

and thus

$$(\text{unbox } c) : \{\text{CoNat}^g \mid \Diamond^{\ell} [Z]\}$$
$$(\text{unbox } m) : \{\text{CoNat}^g \mid \Diamond^{\ell+1} [Z]\}$$
Now, it follows from Ex. E.19 that
\[
\text{fb}^\xi \text{ (unbox } c \text{) (unbox } m) : \{\text{Str}^\xi \text{Bool} \mid \Box^k \bowtie \ell \text{[hd][ff]}\}
\]
so that
\[
\text{box}_i(\text{fb}^\xi \text{ (unbox } c \text{) (unbox } m)) : \Box\{\text{Str}^\xi \text{Bool} \mid \Box^k \bowtie \ell \text{[hd][ff]}\}
\]
Since the formula \(\Box^k \bowtie \ell \text{[hd][ff]}\) is safe we have
\[
\text{box}_i(\text{fb}^\xi \text{ (unbox } c \text{) (unbox } m)) : \{\text{Str Bool} \mid \Box\text{[hd][ff]}\}
\]
The \((\mu)-\text{I})\) rule then gives
\[
\text{box}_i(\text{fb}^\xi \text{ (unbox } c \text{) (unbox } m)) : \{\text{Str Bool} \mid \Box\text{[hd][ff]}\}
\]
and the \((\nu)-\text{I})\) rule gives
\[
\text{box}_i(\text{fb}^\xi \text{ (unbox } c \text{) (unbox } m)) : \{\text{Str Bool} \mid \Box\bowtie\text{[hd][ff]}\}
\]
The result then follows from the fact that
\[
\begin{align*}
\text{Z} & : \{\text{CoNat} \mid \Box\bowtie 1[Z]\} \\
\text{S Z} & : \{\text{CoNat} \mid \Box\bowtie 3[Z]\} \\
\end{align*}
\]
\[
\square
\]

**Example E.21.** We have
\[
\text{fb}^\xi \text{: CoNat}^\xi \rightarrow \{\text{CoNat}^\xi \mid [S]\} \rightarrow \{\text{Str}^\xi \text{Bool} \mid \Box ([\text{hd}][tt] \lor \Box [\text{hd}][tt])\}
\]

**Proof.** Let
\[
T := \{\text{CoNat}^\xi \rightarrow \text{CoNat}^\xi \rightarrow \text{Str}^\xi \text{Bool} \mid \varphi \land \psi\}
\]
\[
\varphi := [S] \rightarrow T \rightarrow [\text{hd}][tt]
\]
\[
\psi := T \rightarrow [S] \rightarrow \Box ([\text{hd}][tt] \lor \Box [\text{hd}][tt])
\]
and assume
\[
g : \triangleright T
\]
Let
\[
M(g, c, m) := \text{case } c \text{ of }
\]
\[
| \text{Z} \rightarrow \text{ff} : \xi \text{g} \odot \text{(next } m) \odot \text{next}(\text{S}^\xi \text{ (next } m))
\]
\[
| \text{S}^\xi n \rightarrow \text{tt} : \xi \text{g} \odot n \odot \text{(next } m)
\]
We show
\[
\lambda c.\lambda m. M(g, c, m) : T
\]
First, by using the \((\text{Inj}_{11}-\text{E})\) rule we easily get
\[
\lambda c.\lambda m. M(g, c, m) : \{\text{CoNat}^\xi \rightarrow \text{CoNat}^\xi \rightarrow \text{Str}^\xi \text{Bool} \mid [S] \rightarrow T \rightarrow [\text{hd}][tt]\}
\]
It remains to show 
\[ \lambda c. \lambda m. M (g, c, m) : \{ \text{CoNat}^{\text{f}} \rightarrow \text{CoNat}^{\text{f}} \rightarrow \text{Str}^{\text{f}} \text{Bool} \mid \top \mid \rightarrow [S] \mid \rightarrow \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \} \]

Assume 
\[ c : \text{CoNat}^{\text{f}} \]
\[ m : \{ \text{CoNat}^{\text{f}} \mid [S] \} \]

We apply the typing rule for case (Fig. 4). This leads to two branches, one for \((\text{unfold } c) = \text{fold} (\text{in } 0 \langle \rangle)\) (denoted \(Z^g\)), and one for \((\text{unfold } c) = \text{fold} (\text{in } 1 n)\) (denoted \(S^g n\)).

**Case of \(Z^g\).**

We have to show 
\[ \text{ff} :^g g \odot (\text{next } m) \odot \text{next}(S \text{ next } m) : \{ \text{Str}^{\text{f}} \text{Bool} \mid \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \} \]

We have 
\[ \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \Leftrightarrow (\text{hd}[tt] \lor \circ \text{hd}[tt]) \land \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \]

and we consider each conjunct separately.

**(Sub)Case of \((\text{hd}[tt] \lor \circ \text{hd}[tt])\).**

Since 
\[ S^g (\text{next } m) : \{ \text{CoNat}^{\text{f}} \mid [S] \} \]
\[ g : \triangleright (\text{CoNat}^{\text{f}} \rightarrow \{ \text{CoNat}^{\text{f}} \mid [S] \} \rightarrow \text{CoNat}^{\text{f}} \rightarrow \{ \text{Str}^{\text{f}} \text{Bool} \mid \text{hd}[tt] \}) \]

we get 
\[ g \odot (\text{next } m) \odot \text{next}(S \text{ next } m) : \triangleright \{ \text{Str}^{\text{f}} \text{Bool} \mid \text{hd}[tt] \} \]

and the result follows.

**(Sub)Case of \(\circ \square (\text{hd}[tt] \lor \circ \text{hd}[tt])\).**

Since 
\[ S^g (\text{next } m) : \{ \text{CoNat}^{\text{f}} \mid [S] \} \]
\[ g : \triangleright (\text{CoNat}^{\text{f}} \rightarrow \{ \text{CoNat}^{\text{f}} \mid [S] \} \rightarrow \text{CoNat}^{\text{f}} \rightarrow \{ \text{Str}^{\text{f}} \text{Bool} \mid \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \}) \]

we get 
\[ g \odot (\text{next } m) \odot \text{next}(S \text{ next } m) : \triangleright \{ \text{Str}^{\text{f}} \text{Bool} \mid \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \} \]

and the result follows.

**Case of \(S^g n\).**

We have to show 
\[ \text{tt} :^g g \odot n \odot (\text{next } m) : \{ \text{Str}^{\text{f}} \text{Bool} \mid \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \} \]

assuming 
\[ n : \text{CoNat}^{\text{f}} \]

We have 
\[ \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \Leftrightarrow (\text{hd}[tt] \lor \circ \text{hd}[tt]) \land \square (\text{hd}[tt] \lor \circ \text{hd}[tt]) \]

and we consider each conjunct separately.
(Sub)Case of \([\text{hd}[tt] \vee \bigcirc \text{hd}[tt]]\).

We have

\[
\text{tt} : \mathcal{G} \oplus n \oplus (\text{next } m) : \{\text{Str}^g \mid \text{hd}[tt]\}
\]

(Sub)Case of \(\bigcirc \bigcirc \bigcirc (\text{hd}[tt] \vee \bigcirc \text{hd}[tt])\).

Since

\[
m : \{\text{CoNat}^g \mid \text{S}\}
\]

\[
g : \triangledown \{\text{CoNat}^g \mid \text{S}\} \rightarrow \{\text{Str}^g \mid \text{hd}[tt] \vee \bigcirc \text{hd}[tt]\}
\]

we get

\[
g \oplus (\text{next } m) \oplus \text{next(S (next } m)) : \triangledown \{\text{Str}^g \mid \text{hd}[tt] \vee \bigcirc \text{hd}[tt]\}
\]

and the result follows.

Example E.22. We have

\[
\text{fb } Z (S Z) : \{\text{Str} \mid \text{box}|\bigcirc \bigcirc \text{hd}[tt]\}
\]

**Proof.** By Ex. E.21 we have

\[
\text{fb}^g (\text{unbox } Z) (\text{unbox (S Z)}) : \{\text{Str}^g \mid \bigcirc (\text{hd}[tt] \vee \bigcirc \text{hd}[tt])\}
\]

so that

\[
\text{fb } Z (S Z) : \triangledown \{\text{Str}^g \mid \bigcirc (\text{hd}[tt] \vee \bigcirc \text{hd}[tt])\}
\]

Since the formula \(\bigcirc (\text{hd}[tt] \vee \bigcirc \text{hd}[tt])\) is safe we get

\[
\text{fb } Z (S Z) : \{\text{Str} \mid \text{box}|\bigcirc (\text{hd}[tt] \vee \bigcirc \text{hd}[tt])\}
\]

Now, the result follows from the fact that

\[
(\text{hd}[tt] \vee \bigcirc \text{hd}[tt]) \Rightarrow \bigcirc \text{hd}[tt]
\]

\(\square\)

The following uses the rule

\[
\vdash_{H \rightarrow A} (\text{ev}(\psi_0)[\varphi] \land \text{ev}(\psi_1)[\varphi]) \Rightarrow \text{ev}(\psi_0 \land \psi_1)[\varphi]
\]

Example E.23. We have

\[
\text{fb}^g : \text{CoNat}^g \rightarrow \{\text{CoNat}^g \mid \text{S}\} \rightarrow \{\text{Str}^g \mid \text{hd}[tt] \vee \bigcirc \text{hd}[tt]\}
\]

**Proof.** Let \(T\) be the type

\[
\{\text{CoNat}^g \rightarrow \text{CoNat}^g \rightarrow \text{Str}^g \mid \text{S}\} \rightarrow \{\text{Str}^g \mid \text{hd}[tt] \land \text{Z} \mid \rightarrow \bigcirc \text{hd}[tt]\}
\]

Note that

\[
T \leq \text{CoNat}^g \rightarrow \{\text{CoNat}^g \mid \text{S}\} \rightarrow \{\text{Str}^g \mid \text{hd}[tt] \vee \bigcirc \text{hd}[tt]\}
\]
Assume \( g : T \)

Let

\[
M(g, c, m) := \begin{cases} 
Z \mapsto & ff \mapsto g \oplus (\text{next} m) \oplus (S g \text{ (next} m)) \\
S \mapsto & tt \mapsto g \oplus n \oplus (\text{next} m)
\end{cases}
\]

We show \( \lambda c. \lambda m. M(g, c, m) : T \)

We consider each conjunct separately.

**Case of** \([S] \mid\rightarrow\top\mid\rightarrow[hd][tt]\).

Assume \( c : \{\text{CoNat}^g \mid [S]\} \)

Applying the (INJ\_1-E) rule, we are done since

\( tt \mapsto g \oplus n \oplus (\text{next} m) : \{\text{Str}^g \text{ Bool} \mid [hd][tt]\} \)

assuming \( n : \text{CoNat}^g \)

**Case of** \([Z] \mid\rightarrow[S] \mid\rightarrow[hd][tt]\).

Assume \( c : \{\text{CoNat}^g \mid [Z]\} \)

\( m : \{\text{CoNat}^g \mid [S]\} \)

Applying the (INJ\_0-E) rule, we are left with showing

\( ff \mapsto g \oplus (\text{next} m) \oplus (S g \text{ (next} m)) : \{\text{Str}^g \text{ Bool} \mid [hd][tt]\} \)

But the result is trivial since

\( g : \{\text{CoNat}^g \rightarrow \text{CoNat}^g \rightarrow \text{Str}^g \text{ Bool} \mid [S] \mid\rightarrow\top\mid\rightarrow[hd][tt]\} \)

\( \square \)

**A Scheduler**

*Example E.24.*

\[
\text{sched} : \text{Str Bool} \rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \text{Str}(A + B)
\]

\[
:= \lambda b. \lambda s. \lambda t. \text{box}_e(\text{sched}^b (\text{unbox} b) (\text{unbox} s) (\text{unbox} t))
\]

\[
\text{sched}^b : \text{Str}^b \text{ Bool} \rightarrow \text{Str}^b A \rightarrow \text{Str}^b B \rightarrow \text{Str}^b(A + B)
\]

\[
:= \text{fix}(g). \lambda b. \lambda s. \lambda t. \text{case} (\text{hd}^b b) \text{ of }
\]

\[
| \text{tt} \mapsto (\text{in}_0 (\text{hd}^b s)) : \text{Str}^b \rightarrow (t^b \text{ b}) \oplus (t^b \text{ s}) \oplus (t^b \text{ t})
\]

\[
| \text{ff} \mapsto (\text{in}_1 (\text{hd}^b t)) : \text{Str}^b \rightarrow (t^b \text{ b}) \oplus (t^b \text{ s}) \oplus (t^b \text{ t})
\]
Example E.25. We can give the following refinement types to \( \{ T \} \):

\[
\begin{align*}
\{ \text{Str Bool} \mid \boxed{\Diamond \mathord{\Box} \text{hd}[|tt]} \} &\rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \{ \text{Str}(A + B) \mid \boxed{\Diamond \mathord{\Box} \text{hd}[|i_{0}]T} \} \\
\{ \text{Str Bool} \mid \boxed{\Diamond \mathord{\Box} \text{hd}[|ff]} \} &\rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \{ \text{Str}(A + B) \mid \boxed{\Diamond \mathord{\Box} \text{hd}[|i_{1}]T} \}
\end{align*}
\]

Proof. Direct, using the following Ex. E.26.

Example E.26. We can give the following refinement types to \( \text{sched} \):

\[
\begin{align*}
\forall k \cdot \forall \ell \cdot \{ \text{Str Bool} \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt] \} &\rightarrow \text{Str} A \rightarrow \text{Str} B \rightarrow \{ \text{Str}(A + B) \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T \} \\
\forall k \cdot \forall \ell \cdot \{ \text{Str Bool} \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|ff] \} &\rightarrow \text{Str A} \rightarrow \text{Str B} \rightarrow \{ \text{Str}(A + B) \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{1}]T \}
\end{align*}
\]

Proof. We only discuss the first type, since the second one is completely similar.
Let \( T(k, \ell) \) be the type

\[
\{ \text{Str Bool} \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt] \} \rightarrow \text{Str} A \rightarrow \text{Str} B \rightarrow \{ \text{Str}(A + B) \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T \}
\]

and assume

\[
g : \vdash \forall k \cdot \forall \ell \cdot T(k, \ell)
\]

Let

\[
M(g, b, s, t) := \text{case } (\text{hd}^\ell b) \text{ of }
\begin{align*}
\text{tt} &\mapsto (i_{0} (\text{hd}^\ell s)) :: \Diamond \mathord{\Box} (t_{0}^\ell b) @ (t_{0}^\ell s) @ (t_{0}^\ell t) \\
\text{ff} &\mapsto (i_{1} (\text{hd}^\ell t)) :: \Diamond \mathord{\Box} (t_{0}^\ell b) @ (t_{0}^\ell s) @ (t_{0}^\ell t)
\end{align*}
\]

We show

\[
\lambda b. \lambda s. \lambda t. M(g, b, s, t) : \forall k \cdot \forall \ell \cdot T(k, \ell)
\]

We apply the \((\forall \text{CI})\) rule on \( \forall k \). In the case of \( \forall \ell \cdot T(0, \ell) \), the result is trivial since

\[
\Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T \leftrightarrow T
\]

As for \( \forall \ell \cdot T(k+1, \ell) \), we apply the \((\forall \text{CI})\) rule, this time on \( \forall \ell \). In the case of \( T(k+1, 0) \), since

\[
\Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt] \leftrightarrow \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt] \land \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt]
\]

we get

\[
\Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt] \leftrightarrow T
\]

and we can conclude using the \((\text{ExF})\) rule. It remains to deal with the case of \( T(k+1, \ell+1) \). We have to show

\[
M(g, b, s, t) : \{ \text{Str}^\ell (A + B) \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T \}
\]

assuming

\[
\begin{align*}
b : \{ \text{Str}^\ell \text{Bool} \mid \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|tt] \} \\
s : \text{Str}^\ell A \\
t : \text{Str}^\ell B
\end{align*}
\]

We have

\[
\Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T \leftrightarrow \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T \land \Diamond \mathord{\Box} \ell \mathord{\Box} \text{hd}[|i_{0}]T
\]

and we consider each conjunct separately.
Case of $\Diamond^{\ell + 1}[\text{hd}][i_{n_0}]\top$.
Since
$$\Box^{\ell + 1}\Diamond^{\ell + 1}[\text{hd}][\text{tt}] \iff \Diamond^{\ell + 1}[\text{hd}][\text{tt}] \land \Box^k \Diamond^{\ell + 1}[\text{hd}][\text{tt}]$$
we have
$$b : \{ \text{Str}^\# \text{Bool} \mid \Diamond^{\ell + 1}[\text{hd}][\text{tt}] \}$$
Using
$$\Diamond^{\ell + 1}[\text{hd}][\text{tt}] \iff [\text{hd}][\text{tt}] \lor \Box^\ell [\text{hd}][\text{tt}]$$
we reason by cases on the refinement type of $b$.
(Sub)Case of $[\text{hd}][\text{tt}]$.
We apply the $(\text{Inj}_0\text{-E})$ rule on $b$ and we are done since
$$(i_{n_0} (\text{hd}^\# s)) : \# g \odot (t_l^\# b) \odot (t_l^\# s) \odot (t_l^\# t) : \{ \text{Str}^\# (A + B) \mid [\text{hd}][i_{n_0}]\top \}$$
(Sub)Case of $\Box^\ell [\text{hd}][\text{tt}]$.
We have
$$t_l^\# b : \triangledown \{ \text{Str}^\# \text{Bool} \mid \Box^\ell [\text{hd}][\text{tt}] \}$$
We apply the case-elimination rule on $b$. In both branches, since (by subtyping) $g$ has type
$$\triangledown \{ \text{Str}^\# (A + B) \mid [\text{hd}][i_{n_0}]\top \}$$
and since, according to Table 2
$$\Box^1 \theta \iff \theta$$
we get
$$g \odot (t_l^\# b) \odot (t_l^\# s) \odot (t_l^\# t) : \triangledown \{ \text{Str}^\# (A + B) \mid \Box^\ell [\text{hd}][i_{n_0}]\top \}$$
so that
$$(-) : \# g \odot (t_l^\# b) \odot (t_l^\# s) \odot (t_l^\# t) : \{ \text{Str}^\# (A + B) \mid \Box^\ell [\text{hd}][i_{n_0}]\top \}$$
and we are done since
$$\Box^\ell [\text{hd}][i_{n_0}]\top \Rightarrow \Diamond^{\ell + 1}[\text{hd}][i_{n_0}]\top$$
Case of $\Box^k \Diamond^{\ell + 1}[\text{hd}][i_{n_0}]\top$.
Since
$$\Box^{\ell + 1}\Diamond^{\ell + 1}[\text{hd}][\text{tt}] \iff \Diamond^{\ell + 1}[\text{hd}][\text{tt}] \lor \Box^k \Diamond^{\ell + 1}[\text{hd}][\text{tt}]$$
we have
$$b : \{ \text{Str}^\# \text{Bool} \mid \Box^k \Diamond^{\ell + 1}[\text{hd}][\text{tt}] \}$$
so that
$$t_l^\# b : \triangledown \{ \text{Str}^\# \text{Bool} \mid \Box^k \Diamond^{\ell + 1}[\text{hd}][\text{tt}] \}$$
We apply the case-elimination rule on \( b \). In both branches, since (by subtyping) \( g \) has type

\[
\lhd \left( \{ \text{Str}^g \text{Bool} \mid \square^k \Diamond^{\ell+1}[\text{hd}][\text{tt}] \} \rightarrow \text{Str}^g A \rightarrow \text{Str}^g B \rightarrow \{ \text{Str}^g (A + B) \mid \square^k \Diamond^{\ell+1}[\text{hd}][\text{in}_0] \top \} \right)
\]

we get

\[
g \odot (\text{tl}^g b) \odot (\text{tl}^g s) \odot (\text{tl}^g t) : \square^k \Diamond^{\ell+1}[\text{hd}][\text{in}_0] \top
\]

so that

\[
\bot :: g \odot (\text{tl}^g b) \odot (\text{tl}^g s) \odot (\text{tl}^g t) : \square^k \Diamond^{\ell+1}[\text{hd}][\text{in}_0] \top
\]

\[
\square
\]

### E.6 Colists

We detail here the refinement types given to the guarded and coinductive append functions on colists in Table 4. We present some basic material in §E.6. The append function itself is detailed in §E.6, and we give sharper refinements with iteration terms in §E.6. We begin in §E.6 with an overview of the main examples on colists.

**Overview** The cases of

\[
\text{append}^g : \{ \text{CoList}^g A \mid [\neg\text{nil}] \} \rightarrow \text{CoList}^g A \rightarrow \{ \text{CoList}^g A \mid [\neg\text{nil}] \}
\]

\[
\text{append}^g : \text{CoList}^g A \rightarrow \{ \text{CoList}^g A \mid [\neg\text{nil}] \} \rightarrow \{ \text{CoList}^g A \mid [\neg\text{nil}] \}
\]

are detailed in Ex. E.33.

We now discuss

\[
\text{append} : \{ \text{CoList} A \mid [\text{box}][\text{fin}] \} \rightarrow \{ \text{CoList} A \mid [\text{box}][\text{fin}] \} \rightarrow \{ \text{CoList} A \mid [\text{box}][\text{fin}] \}
\]

which says that append takes finite colists to a finite colist. Recall that \([\text{fin}] = \Diamond^{3}[\text{nil}]\). Details are given in Ex. E.35. The other refinement types for append are detailed in Ex. E.36 and Ex. E.37.

We refer here to the code of the append function as defined in Table 3 and Ex. E.32. First, since \(\Diamond^{3}[\text{nil}]\) is smooth, we can apply the rule (\(\mu\)-E) (Fig. 11) twice and reduce to

\[
E \vdash \text{box},(\text{append}^g (\text{unbox} s) (\text{unbox} t)) : \{ \text{CoList} A \mid [\text{box}]\Diamond[\text{nil}] \}
\]

where \( E \) assumes \( s \) of type \( \{ \text{CoList} A \mid [\text{box}]\Diamond^k[\text{nil}] \} \) and \( t \) of type \( \{ \text{CoList} A \mid [\text{box}]\Diamond^\ell[\text{nil}] \} \).

Using the derived rule (\(\mu\)-I) (Ex. 6.10), we further reduce to

\[
E \vdash \text{box},(\text{append}^g (\text{unbox} s) (\text{unbox} t)) : \{ \text{CoList} A \mid [\text{box}]\Diamond^{k+\ell}[\text{nil}] \}
\]

Now, since the formulae \(\Diamond^k[\text{nil}]\) are safe, by subtyping (Fig. 11) we have

\[
E \vdash s : \llbracket \text{CoList} A \mid \Diamond^k[\text{nil}] \rrbracket \quad \text{and} \quad E \vdash t : \llbracket \text{CoList} A \mid \Diamond^\ell[\text{nil}] \rrbracket
\]
and we can reduce to showing that the guarded append\(^7\) has type \(\forall k \cdot \forall \ell \cdot T(k, \ell)\),
where
\[
T(k, \ell) := \{\text{CoList}^g A \mid \Diamond^k \text{nil} \} \rightarrow \{\text{CoList}^g A \mid \Diamond^\ell \text{nil} \} \rightarrow \{\text{CoList}^g A \mid \Diamond^{k+\ell} \text{nil} \}
\]
Let \(N(g, s, t)\) be such that append\(^g\) = \text{fix}(g).\lambda s.\lambda t. N(g, s, t).\) We show
\[
\lambda s.\lambda t. N(g, s, t) : \forall k \cdot \forall \ell \cdot T(k, \ell)
\]
in a typing context (leaved implicit) which assumes \(g\) of type \(\forall k \cdot \forall \ell \cdot T(k, \ell)\). We apply the \((\forall\text{-CI})\) rule on \(\forall k \cdot \forall \ell \cdot T(k, \ell)\). Since \(\Diamond^0 \text{nil} \Leftrightarrow \bot\), the branch of \(\forall \ell \cdot T(0, \ell)\) can be dealt with using the \((\text{ExF})\) rule. In the branch of \(\forall \ell \cdot T(k+1, \ell)\), we apply the \((\forall\text{-I})\) rule. We are thus left with showing
\[
N(g, s, t) : \{\text{CoList}^g A \mid \Diamond^{k+\ell+1} \text{nil} \}
\]
assuming further \(s : \{\text{CoList}^g A \mid \Diamond^{k+\ell+1} \text{nil} \}\) and \(t : \{\text{CoList}^g A \mid \Diamond^\ell \text{nil} \}\). We unfold \(\Diamond^{k+\ell+1} \text{nil} \)
as
\[
\Diamond^{k+\ell+1} \text{nil} \Rightarrow [\text{nil}] \lor \Diamond^k \text{nil}
\]
Using the \((\forall\text{-E})\) rule, we have two cases for the refinement type of \(s\). In the case of \(\{\text{CoList} A \mid [\text{nil}]\}\), since \([\text{nil}] = [\text{fold}][\text{in}_0]T\), we have \((\text{unfold} s) : [\text{in}_0]T\). Thanks to the \((\text{INJ}_0)\) rule, we are left with showing
\[
t : \{\text{CoList} A \mid \Diamond^\ell \text{nil} \} \vdash t : \{\text{CoList} A \mid \Diamond^{k+\ell} \text{nil} \}
\]
But we are done since \([\ell] \leq [k+\ell+1]\) so that
\[
\Diamond^\ell \text{nil} \Rightarrow \Diamond^{k+\ell +1} \text{nil}
\]
Assume now that \(s\) has type \(\{\text{CoList} A \mid \Diamond^k \text{nil} \}\). By unfolding \(\Diamond^{k+\ell+1} \text{nil} \) we reduce to showing
\[
N(g, s, t) : \{\text{CoList}^g A \mid \Diamond^k \text{nil} \}
\]
Since, on colists, \(\Diamond(-) = [\text{fold}][\text{in}_1][\pi_1][\text{next}](\cdot)\), we can apply the \((\text{INJ}_1\text{-E})\) rule on \((\text{unfold} s)\). This amounts to showing
\[
\text{Cons}^g \ x \ (g \oplus x \oplus (\text{next } t)) : \{\text{CoList} A \mid \Diamond^k \text{nil} \}
\]
where, since
\[
(\text{unfold } s) : \{1 + A \times \Diamond^k \text{nil} \}
\]
we can assume \(x s : \Diamond^k \text{nil} \). By subtyping and \((\forall\text{-E})\) we have \(g : \Diamond^k \text{nil} \), so that
\[
g \oplus x \oplus (\text{next } t) : \Diamond^k \text{nil}
\]
and we conclude by the analogue of Ex. 5.3 for colists. The other typings for append\(^g\) are dealt with similarly. Let us finally just mention that the type of append\(^g\) can be sharpened to
\[
\forall k \cdot \forall \ell \cdot \{\text{CoList}^g A \mid \Diamond^k \text{nil} \} \rightarrow \{\text{CoList}^g A \mid \Diamond^{\ell+1} \text{nil} \} \rightarrow \{\text{CoList}^g A \mid \Diamond^{k+\ell} \text{nil} \}\)
\]
reflecting that on finite colists, append\(^g\) removes one constructor Nil\(^g\) from its arguments (see Ex. E.38).
The Type of CoLists

The type of colists is

\[
\text{CoList} \ A := \Box \text{CoList}^\delta \ A \\
\text{CoList}^\delta \ A := \text{Fix}(X).1 + A \times \rightarrow X
\]

Its usual guarded constructors are represented as

\[
\begin{align*}
\text{Nil}^\delta & := \text{fold}(\text{in}_0()) : \text{CoList}^\delta \ A \\
\text{Cons}^\delta & := \lambda x.\lambda xs.\text{fold}(\text{in}_1(x,xs)) : A \rightarrow \rightarrow \text{CoList}^\delta \ A \rightarrow \text{CoList}^\delta \ A
\end{align*}
\]

Their coinductive (for \(A\) a constant type) variants are

\[
\begin{align*}
\text{Nil} & := \text{box}_i(\text{Nil}^\delta) : \text{CoList} \ A \\
\text{Cons} & := \lambda x.\lambda xs.\text{box}_i(\text{Cons}^\delta x (\text{next (unbox xs)})) : A \rightarrow \text{CoList} \ A \rightarrow \text{CoList} \ A
\end{align*}
\]

Note E.27. Extending the notation for (guarded) streams, we often write

\[
\begin{align*}
(x ::^\delta xs) & := \text{Cons}^\delta x xs \\
(x :: xs) & := \text{Cons} x xs
\end{align*}
\]

\[
\begin{align*}
\text{ Nil } & := \text{Nil}^\delta \\
\text{Cons} & := \lambda x.\lambda xs.\text{Nil}(\text{Cons}^\delta x (\text{next (unbox xs)})) : A \rightarrow \text{CoList} \ A \rightarrow \text{CoList} \ A
\end{align*}
\]

Note E.28 (Syntactic Sugar for Pattern Matching). Assuming \(s : \text{CoList}^\delta \ A\), we often write

\[
\begin{align*}
\text{ case } s \text{ of } \\
| \text{ Nil}^\delta & \mapsto N \\
| \text{ Cons}^\delta x xs & \mapsto M
\end{align*}
\]

for

\[
\begin{align*}
\text{ case } (\text{unfold } s) \text{ of } \\
| y. N[()]/y & \\
| y. M[\pi_0(y)/x, \pi_1(y)/xs]
\end{align*}
\]

Example E.29 (Formulae over \text{CoList}^\delta). Assuming \(\psi : A\) and \(\varphi : \text{CoList}^\delta \ A\),

\[
\begin{align*}
[\text{nil}] & := \text{fold}(\text{in}_0) : \top : \text{CoList}^\delta \ A \\
[\neg\text{nil}] & := \text{fold}(\text{in}_1) : \top : \text{CoList}^\delta \ A \\
[\text{hd}\psi] & := \text{fold}(\text{in}_1[\pi_0]\varphi) : \text{CoList}^\delta \ A \\
\square\varphi & := \mu \alpha.\varphi \land \square\alpha : \text{CoList}^\delta \ A \\

\square\lor\varphi & := \mu \alpha.\varphi \lor \square\alpha : \text{CoList}^\delta \ A \\
\square\land\varphi & := \nu \alpha.\varphi \land \square\alpha : \text{CoList}^\delta \ A \\
[\text{inf}] & := \square[\neg\text{nil}] : \text{CoList}^\delta \ A \\
[\text{fin}] & := \Diamond[\neg\text{nil}] : \text{CoList}^\delta \ A
\end{align*}
\]

Example E.30.

\[
\begin{align*}
\text{Cons}^\delta & : A \rightarrow \rightarrow \text{CoList}^\delta \ A \rightarrow \{\text{CoList}^\delta \ A \mid [\neg\text{nil}]\} \\
\text{Cons} & : A \rightarrow \rightarrow \{\text{CoList}^\delta \ A \mid \text{inf}\} \rightarrow \{\text{CoList}^\delta \ A \mid \text{inf}\} \\
\text{Nil} & : \{\text{CoList}^\delta \ A \mid \text{nil}\}
\end{align*}
\]
Note that
\[ \vdash \text{CoList} A [\text{nil}] \Rightarrow \square \text{fin} \varphi \]

**Example E.31.** Similarly as in §E.1 and §E.1, assuming \( \varphi : A \) we have

\[
\begin{align*}
\text{Cons}^g &: \{ A \mid \varphi \} \rightarrow \nrightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \} \\
\text{Cons}^g &= \{ A \mid \varphi \} \rightarrow \nrightarrow \{ \text{CoList}^g A \mid \text{nil} \} \rightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \}
\end{align*}
\]

\[
\begin{align*}
\text{Nil}^g &: \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \}
\end{align*}
\]

\[
\begin{align*}
\text{Cons}^g &: \{ A \mid \varphi \} \rightarrow \nrightarrow \{ \text{CoList}^g A \mid \langle \text{hd} \rangle \varphi \} \rightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \}
\end{align*}
\]

**The Append Function on Colists**

**Example E.32 (The Append Function on Colists).**

\[
\begin{align*}
\text{append}^g &: \text{CoList}^g A \rightarrow \text{CoList}^g A \rightarrow \text{CoList}^g A \\
:=& \text{fix}(g) \cdot \lambda s. \lambda t. \text{case } s \text{ of} \\
& \mid \text{Nil}^g \rightarrow t \\
& \mid \text{Cons}^g x x s \rightarrow \text{Cons}^g x (g \circ x \circ (\text{next} t))
\end{align*}
\]

\[
\begin{align*}
\text{append} &: \text{CoList} A \rightarrow \text{CoList} A \rightarrow \text{CoList} A \\
:=& \lambda s. \lambda t. \text{box } \iota (\text{append}^g (\text{unbox} s) (\text{unbox} t))
\end{align*}
\]

**Example E.33 (Properties of Append).**

\[
\begin{align*}
\text{append}^g &: \{ \text{CoList}^g A \mid \neg \text{nil} \} \rightarrow \text{CoList}^g A \rightarrow \text{CoList}^g A \\
& \rightarrow \{ \text{CoList}^g A \mid \neg \text{nil} \}
\end{align*}
\]

\[
\begin{align*}
\text{append}^g &: \text{CoList}^g A \rightarrow \{ \text{CoList}^g A \mid \neg \text{nil} \} \rightarrow \{ \text{CoList}^g A \mid \neg \text{nil} \}
\end{align*}
\]

**Example E.34.** Assuming \( \varphi : A \),

\[
\begin{align*}
\text{append}^g &: \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \} \rightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \} \rightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \}
\end{align*}
\]

**Proof.** Let

\[
T := \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \} \rightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \} \rightarrow \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \}
\]

and assume

\[
\begin{align*}
g &: \nrightarrow T \\
s &: \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \} \\
t &: \{ \text{CoList}^g A \mid \square \text{fin} \langle \text{hd} \rangle \varphi \}
\end{align*}
\]

Note that

\[
\square \text{fin} \langle \text{hd} \rangle \varphi \leftrightarrow [\text{nil}] \vee (\langle \text{hd} \rangle \varphi \land \square \text{fin} \langle \text{hd} \rangle \varphi)
\]

We reason by cases on the refinement type of \( s \), applying the \((\lor \rightarrow \lor)\) rule (Fig. 5).
Case of \([\mathbb{nil}]\).
We thus have

\[
\text{unfold}(s) : \{ 1 + A \times ▶ \text{CoList}^S A \mid [\mathbb{in}_0] \top \}
\]

We apply the \((\text{INj}_0^-\text{-E})\) rule and get the result by

\[
t : \{ \text{CoList}^S A \mid □^\text{fin}[\text{hd}]\varphi \}
\]

Case of \([\text{hd}]\varphi \land □^\text{fin}[\text{hd}]\varphi\).
We thus have

\[
s : \{ \text{CoList}^S A \mid [\text{hd}]\varphi \land □^\text{fin}[\text{hd}]\varphi \}
\]

Since the modalities \([\text{fold}]\) and \([\text{in}_1]\) preserve \land this gives

\[
s : \{ \text{CoList}^S A \mid [\text{fold}][\text{in}_1] \left( ([\pi_0]\varphi \land [\pi_1][\text{next}][□^\text{fin}[\text{hd}]\varphi] \right) \}
\]

so that

\[
\text{unfold}(s) : \{ 1 + A \times ▶ \text{CoList}^S A \mid [\text{in}_1] \left( ([\pi_0]\varphi \land [\pi_1][\text{next}][□^\text{fin}[\text{hd}]\varphi] \right) \}
\]

We then apply the \((\text{INj}_1^-\text{-E})\) rule. Assume

\[
y : \{ A \times ▶ \text{CoList}^S A \mid [\pi_0]\varphi \land [\pi_1][\text{next}][□^\text{fin}[\text{hd}]\varphi] \}
\]

and let

\[
x := \pi_0(y) : \{ A \mid \varphi \}
\]

\[
xs := \pi_1(y) : ▶ \{ \text{CoList}^S A \mid □^\text{fin}[\text{hd}]\varphi \}
\]

Then Ex. \(E.31\) easily gives

\[
\text{Cons}^S x (g \oplus xs \oplus (\text{next} t)) : \{ \text{CoList}^S A \mid □^\text{fin}[\text{hd}]\varphi \}
\]

\(\square\)

Example \(E.35\).
append : \{ \text{CoList} A \mid \mathbb{box} \diamond [\mathbb{nil}] \} \rightarrow \{ \text{CoList} A \mid \mathbb{box} \diamond [\mathbb{nil}] \} \rightarrow \{ \text{CoList} A \mid \mathbb{box} \diamond [\mathbb{nil}] \}
append^S : \forall k \cdot \forall \ell \cdot (\{ \text{CoList}^S A \mid \diamond^k [\mathbb{nil}] \} \rightarrow \{ \text{CoList}^S A \mid \diamond^\ell [\mathbb{nil}] \} \rightarrow \{ \text{CoList}^S A \mid \diamond^{k+\ell} [\mathbb{nil}] \})

Proof. Let

\[
T(k, \ell) := (\{ \text{CoList}^S A \mid \diamond^k [\mathbb{nil}] \} \rightarrow \{ \text{CoList}^S A \mid \diamond^\ell [\mathbb{nil}] \} \rightarrow \{ \text{CoList}^S A \mid \diamond^{k+\ell} [\mathbb{nil}] \})
\]

and assume

\[
g : ▶ \forall k \cdot \forall \ell . T(k, \ell)
\]

Let

\[
M(g, s, t) := \text{case } s \text{ of }
\]

\[
\begin{align*}
&\text{Nil}^S \mapsto t \\
&\text{Cons}^S x xs \mapsto \text{Cons}^S x (g \oplus xs \oplus (\text{next} t))
\end{align*}
\]

We show

\[
\lambda s. \lambda t. M(g, s, t) : \forall k \cdot \forall \ell . T(k, \ell)
\]

We apply the \((\forall\text{-CI})\) rule on \(\forall k\). This leads to two cases.
Case of $\forall \ell \cdot T(0, \ell)$.
Apply the ($\forall$-I) rule on $\forall \ell$ and assume

$$s : \{ \text{CoList}^g A \mid \Diamond^0[\text{nil}] \}$$

Since

$$\Diamond^0[\text{nil}] \iff \bot$$

the result follows using the rule (ExF).

Case of $\forall \ell \cdot T(k+1, \ell)$.
Apply the ($\forall$-I) rule on $\forall \ell$ and assume

$$s : \{ \text{CoList}^g A \mid \Diamond^{k+1}[\text{nil}] \}$$
$$t : \{ \text{CoList}^g A \mid \Diamond^\ell[\text{nil}] \}$$

We have to show

$$M(g, s, t) : \{ \text{CoList}^g A \mid \Diamond^{k+1+\ell}[\text{nil}] \}$$

Using

$$\Diamond^{k+1}[\text{nil}] \iff [\text{nil}] \lor \Diamond^k[\text{nil}]$$

we apply the ($\lor$-E) rule on the refinement type of $s$. This leads to two subcases.

(Sub)Case of $[\text{nil}]$.
We have

$$\text{unfold}(s) : \{ (1 + A \times \text{CoList}^g A \mid [\text{in}_0] \top) \}$$

Since $[\ell] \leq [k+1 + \ell]$, the result then follows by applying the (Inj$_0$-E) rule.

(Sub)Case of $\Diamond^k[\text{nil}]$.
We have

$$\text{unfold}(s) : \{ (1 + A \times \text{CoList}^g A \mid [\text{in}_1][\pi_1][\text{next}] \Diamond^k[\text{nil}] \}$$

Using the (Inj$_1$-E) rule we are left with showing

$$\text{Cons}^g x (g \odot xs \odot (\text{next} t)) : \{ \text{CoList}^g A \mid \Diamond^{(k+\ell)+1}[\text{nil}] \}$$

where

$$x := \pi_0(y) : A$$
$$xs := \pi_1(y) : \text{CoList}^g A \mid \Diamond^k[\text{nil}]$$

assuming

$$y : \{ A \times \text{CoList}^g A \mid [\pi_1][\text{next}] \Diamond^k[\text{nil}] \}$$

We have

$$g \odot xs \odot (\text{next} t) : \text{CoList}^g A \mid \Diamond^{k+\ell}[\text{nil}]$$

It follows that

$$\text{Cons}^g x (g \odot xs \odot (\text{next} t)) : \{ \text{CoList}^g A \mid \Diamond^{k+\ell}[\text{nil}] \}$$

and we are done since

$$\Diamond^{k+\ell}[\text{nil}] \Rightarrow \Diamond^{(k+\ell)+1}[\text{nil}]$$
Example E.36. Assuming $\phi : A$,

\[
\text{append} : \{\text{CoList } A | [\Box] \Diamond [\text{hd}] \phi \} \longrightarrow \text{CoList } A \longrightarrow \{\text{CoList } A | [\Box] \Diamond [\text{hd}] \phi \}
\]

\[
\text{append}^k : \forall k \cdot \{\{\text{CoList}^k A | \Diamond^k [\text{hd}] \phi \} \longrightarrow \text{CoList}^k A \longrightarrow \{\text{CoList}^k A | \Diamond^k [\text{hd}] \phi \}\}
\]

where, in the case of $\text{append}$, $\phi : A$ is safe and smooth.

Proof. Let

\[
T(k) := \{\text{CoList}^k A | \Diamond^k [\text{hd}] \phi \} \longrightarrow \text{CoList}^k A \longrightarrow \{\text{CoList}^k A | \Diamond^k [\text{hd}] \phi \}
\]

and assume

\[
g : \triangleright \forall k \cdot T(k)
\]

Let

\[
M(g, s, t) := \text{case } s \text{ of } \begin{array}{l}
\text{Nil}^k \mapsto t \\
\text{Cons}^k x xs \mapsto \text{Cons}^k x (g \odot xs \odot (\text{next } t))
\end{array}
\]

We show

\[
\lambda s. \lambda t. M(g, s, t) : \forall k \cdot T(k)
\]

We apply the ($\forall$-CI) rule on $\forall k$. This leads to two cases.

Case of $T(0)$.
Assume

\[
s : \{\text{CoList}^k A | \Diamond^0 [\text{hd}] \phi \}
\]

Since

\[
\Diamond^0 [\text{hd}] \phi \leftrightarrow \bot
\]

the result follows using the rule (ExF).

Case of $T(k+1)$.
Assume

\[
s : \{\text{CoList}^k A | \Diamond^{k+1} [\text{hd}] \phi \}
\]

\[
t : \text{CoList}^k A
\]

Using

\[
\Diamond^{k+1} [\text{hd}] \phi \leftrightarrow [\text{hd}] \phi \lor \Diamond^k [\text{hd}] \phi
\]

we apply the ($\lor$-E) rule on the refinement type of $s$. This leads to two subcases.

(Sub)Case of $[\text{hd}] \phi$.
We have

\[
\text{unfold}(s) : \{1 + A \times [\text{in}_1] [\pi_0] \phi \}
\]

Using the ($\text{InJ}_1$-E) rule we are left with showing

\[
\text{Cons}^k x (g \odot xs \odot (\text{next } t)) : \{\text{CoList}^k A | \Diamond^{k+1} [\text{hd}] \phi \}
\]

where

\[
x := \pi_0(y) : \{A | \phi \}
\]

\[
xs := \pi_1(y) : \triangleright \text{CoList}^k A
\]
assuming

\[ y : \{ A \times \mathbf{CoList}^g A \mid [\pi_0]\varphi \} \]

We have

\[\text{Cons}^g x (g \oplus xs \oplus (\text{next } t)) : \{ \text{CoList}^g A \mid [\text{hd}]\varphi \}\]

and we are done since

\[[\text{hd}]\varphi \Rightarrow \mathcal{O}^{k+1}[\text{hd}]\varphi\]

(Sub)Case of \( \bigcirc \mathcal{O}^k[\text{hd}]\varphi \).

We have

\[\text{unfold}(s) : \{ 1 + A \times \mathbf{CoList}^g A \mid [\text{in}_1][\pi_1][\text{next}]\mathcal{O}^k[\text{hd}]\varphi \}\]

Using the (IN1-E) rule we are left with showing

\[\text{Cons}^g x (g \oplus xs \oplus (\text{next } t)) : \{ \text{CoList}^g A \mid \mathcal{O}^{k+1}[\text{hd}]\varphi \}\]

where

\[x := \pi_0(y) : A\]
\[xs := \pi_1(y) : \bigcirc \{ \text{CoList}^g A \mid \mathcal{O}^k[\text{hd}]\varphi \}\]

assuming

\[y : \{ A \times \mathbf{CoList}^g A \mid [\pi_1][\text{next}]\mathcal{O}^k[\text{hd}]\varphi \}\]

We have

\[g \oplus xs \oplus (\text{next } t) : \bigcirc \{ \text{CoList}^g A \mid \mathcal{O}^k[\text{hd}]\varphi \}\]

It follows that

\[\text{Cons}^g x (g \oplus xs \oplus (\text{next } t)) : \{ \text{CoList}^g A \mid \bigcirc \mathcal{O}^k[\text{hd}]\varphi \}\]

and we are done since

\[\bigcirc \mathcal{O}^k[\text{hd}]\varphi \Rightarrow \mathcal{O}^{k+1}[\text{hd}]\varphi\]

\[\square\]

Example E.37. Assuming \( \varphi : A \), we have

\[\text{append} : \{ \text{CoList} A \mid [\text{box}]\mathcal{O}[\text{nil}] \} \rightarrow \{ \text{CoList} A \mid [\text{box}]\mathcal{O}[\text{hd}]\varphi \} \rightarrow \{ \text{CoList} A \mid [\text{box}]\mathcal{O}[\text{hd}]\varphi \}\]

\[\text{append}^g : \forall k \cdot \forall \ell \cdot (\{ \text{CoList}^g A \mid \mathcal{O}^k[\text{nil}] \} \rightarrow \{ \text{CoList}^g A \mid \mathcal{O}^\ell[\text{hd}]\varphi \} \rightarrow \{ \text{CoList}^g A \mid \mathcal{O}^{k+\ell}[\text{hd}]\varphi \})\]

where, in the case of \text{append}, \( \varphi : A \) is safe and smooth.
Proof. Let
\[ T(k, \ell) := (\{ \text{CoList}^k A | \diamond^k[\text{nil}] \} \rightarrow \{ \text{CoList}^k A | \diamond^f[\text{hd}]\varphi \} \rightarrow \{ \text{CoList}^k A | \diamond^{k+\ell}[\text{hd}]\varphi \}) \]
and assume
\[ g : \triangleright \forall k \cdot \forall \ell \cdot T(k, \ell) \]
Let
\[ M(g, s, t) := \text{case } s \text{ of } \]
\[ \mid \text{Nil} \mapsto t \]
\[ \mid \text{Cons}^k x x s \mapsto \text{Cons}^k x (g \odot xs \odot (\text{next } t)) \]
We show
\[ \lambda s. \lambda t. M(g, s, t) : \forall k \cdot \forall \ell \cdot T(k, \ell) \]
We apply the (\forall-CI) rule on \forall k. This leads to two cases.

Case of \forall \ell \cdot T(0, \ell).
We apply the (\forall-I) rule on \forall \ell and assume
\[ s : \{ \text{CoList}^k A | \diamond^0[\text{nil}] \} \]
Since
\[ \diamond^0[\text{nil}] \iff \bot \]
the result follows using the rule (EXF).

Case of \forall \ell \cdot T(k+1, \ell).
We apply the (\forall-I) rule on \forall \ell and assume
\[ s : \{ \text{CoList}^k A | \diamond^{k+1}[\text{nil}] \} \]
\[ t : \{ \text{CoList}^k A | \diamond^f[\text{hd}]\varphi \} \]
Using
\[ \diamond^{k+1}[\text{nil}] \iff [\text{nil}] \lor \odot^k[\text{nil}] \]
we apply the (\lor-E) rule on the refinement type of s. This leads to two subcases.

(Sub) Case of [\text{nil}].. We have
\[ \text{unfold}(s) : \{ 1 + A \times \triangleright \text{CoList}^k A | [\text{in}_0] \top \} \]
Since \[ [\ell] \leq [k + 1 + \ell] \], the result then follows by applying the (INJ0-E) rule.

(Sub) Case of \odot^k[\text{nil}].. We have
\[ \text{unfold}(s) : \{ 1 + A \times \triangleright \text{CoList}^k A | [\text{in}_1][\text{in}_2][\text{next}][\diamond^k[\text{nil}]] \} \]
Using the (INJ1-E) rule we are left with showing
\[ \text{Cons}^k x (g \odot xs \odot (\text{next } t)) : \{ \text{CoList}^k A | \diamond^{(k+\ell)+1}[\text{hd}]\varphi \} \]
where
\[ x := \pi_0(y) : A \]
\[ xs := \pi_1(y) : ▶ \{ \text{CoList}^g A \mid □^k[\text{nil}] \} \]
assuming
\[ y : \{ A \times ▶ \text{CoList}^g A \mid [\pi_1][\text{next}] □^k[\text{nil}] \} \]

We have
\[ g \odot xs \odot (\text{next } t) : ▶ \{ \text{CoList}^g A \mid □^{k+\ell}[\text{hd}][\varphi] \} \]

It follows that
\[ \text{Cons}^g x (g \odot xs \odot (\text{next } t)) : \{ \text{CoList}^g A \mid □ □^{k+\ell}[\text{hd}][\varphi] \} \]
and we are done since
\[ □ □^{k+\ell}[\text{hd}][\varphi] \Rightarrow □^{(k+\ell)+1}[\text{hd}][\varphi] \]

Sharper Refinements for the Append Function on Colists

*Example E.38.*

\[ \text{append}^g : \forall k \cdot \forall \ell \cdot \left( \{ \text{CoList}^g A \mid □^k[\text{nil}] \} \rightarrow \{ \text{CoList}^g A \mid □^{\ell+1}[\text{nil}] \} \rightarrow \{ \text{CoList}^g A \mid □^{k+\ell}[\text{nil}] \} \right) \]

**Proof.** Let
\[ T(k, \ell) := \left( \{ \text{CoList}^g A \mid □^k[\text{nil}] \} \rightarrow \{ \text{CoList}^g A \mid □^{\ell+1}[\text{nil}] \} \rightarrow \{ \text{CoList}^g A \mid □^{k+\ell}[\text{nil}] \} \right) \]
and assume
\[ g : ▶ \forall k \cdot \forall \ell \cdot T(k, \ell) \]

Let
\[ M(g, s, t) := \text{case } s \text{ of} \]
\[ \mid \text{Nil}^g \mapsto t \]
\[ \mid \text{Cons}^g x xs \mapsto \text{Cons}^g x (g \odot xs \odot (\text{next } t)) \]

We show
\[ \lambda s.\lambda t.M(g, s, t) : \forall k \cdot \forall \ell \cdot T(k, \ell) \]

We apply the (∀-CI) rule on ∀k. This leads to two cases.

**Case of ∀k \cdot T(0, \ell).**

Apply the (∀-I) rule on ∀t and assume
\[ s : \{ \text{CoList}^g A \mid □^0[\text{nil}] \} \]

Since
\[ □^0[\text{nil}] \iff \bot \]
the result follows using the rule (ExF).
Case of $\forall \ell \cdot T(k+1, \ell)$.

Apply the ($\forall$-I) rule on $\forall \ell$ and assume

$$s : \{\text{CoList}^g A \mid \Diamond^{k+1}[\text{nil}]\}$$
$$t : \{\text{CoList}^g A \mid \Diamond^{\ell+1}[\text{nil}]\}$$

We have to show

$$M(g, s, t) : \{\text{CoList}^g A \mid \Diamond^{k+1+\ell}[\text{nil}]\}$$

Using

$$\Diamond^{k+1}[\text{nil}] \iff [\text{nil}] \lor \Diamond^k[\text{nil}]$$
we apply the ($\lor$-E) rule on the refinement type of $s$. This leads to two subcases.

(Sub)Case of $[\text{nil}]$.

We have

$$\text{unfold}(s) : \{1 + A \times \triangleright \text{CoList}^g A \mid [\text{in}_0] \top\}$$

Since $[\ell+1] \leq [k+1+\ell]$, the result then follows by applying the (INJ$_0$-E) rule.

(Sub)Case of $\Diamond^k[\text{nil}]$.

We have

$$\text{unfold}(s) : \{1 + A \times \triangleright \text{CoList}^g A \mid [\text{in}_1][\text{/init}][\pi_1][\text{next}]\Diamond^k[\text{nil}]\}$$

Using the (INJ$_1$-E) rule we are left with showing

$$\text{Cons}^g \ x (g \odot xs \odot (\text{next} \ t)) : \{\text{CoList}^g A \mid \Diamond^{k+1+\ell}[\text{nil}]\}$$

where

$$x := \pi_0(y) : A$$
$$xs := \pi_1(y) : \triangleright \{\text{CoList}^g A \mid \Diamond^k[\text{nil}]\}$$

assuming

$$y : \{A \times \triangleright \text{CoList}^g A \mid [\pi_1][\text{next}]\Diamond^k[\text{nil}]\}$$

We have

$$g \odot xs \odot (\text{next} \ t) : \triangleright \{\text{CoList}^g A \mid \Diamond^{k+\ell}[\text{nil}]\}$$

It follows that

$$\text{Cons}^g \ x (g \odot xs \odot (\text{next} \ t)) : \{\text{CoList}^g A \mid \Diamond^{k+\ell}[\text{nil}]\}$$

and we are done since

$$\Diamond^{k+\ell}[\text{nil}] \Rightarrow \Diamond^{k+1+\ell}[\text{nil}]$$

Example E.39. Assuming $\varphi : A$, we have

$$\text{append}^g : \forall k \cdot \forall \ell \cdot (\{\text{CoList}^g A \mid \Diamond^k[\text{nil}]\} \rightarrow \{\text{CoList}^g A \mid \Diamond^{\ell+1}[\text{hd}\varphi]\} \rightarrow \{\text{CoList}^g A \mid \Diamond^{k+\ell}[\text{hd}\varphi]\})$$
Proof. Let
\[ T(k, \ell) := \{ \text{CoList}^k A \mid \Diamond^k \text{[nil]} \} \rightarrow \{ \text{CoList}^\ell A \mid \Diamond^{\ell+1} \text{[hd][\varphi]} \} \rightarrow \{ \text{CoList}^k A \mid \Diamond^{k+\ell} \text{[hd][\varphi]} \} \]
and assume
\[ g : \forall k \cdot \forall \ell \cdot T(k, \ell) \]
Let
\[ M(g, s, t) := \text{case } s \text{ of} \]
\[ | \text{Nil}^\ell \mapsto t \]
\[ | \text{Cons}^\ell x xs \mapsto \text{Cons}^\ell x (g \odot xs \odot (\text{next } t)) \]
We show
\[ \lambda s. \lambda t. M(g, s, t) : \forall k \cdot \forall \ell \cdot T(k, \ell) \]
We apply the (\forall-\text{CI}) rule on \forall k. This leads to two cases.

**Case of \forall \ell \cdot T(0, \ell).**
We apply the (\forall-\text{I}) rule on \forall \ell and assume
\[ s : \{ \text{CoList}^\ell A \mid \Diamond^0 \text{[nil]} \} \]
Since
\[ \Diamond^0 \text{[nil]} \Leftrightarrow \bot \]
the result follows using the rule (EXF).

**Case of \forall \ell \cdot T(k+1, \ell).**
We apply the (\forall-\text{I}) rule on \forall \ell and assume
\[ s : \{ \text{CoList}^\ell A \mid \Diamond^{k+1} \text{[nil]} \} \]
\[ t : \{ \text{CoList}^\ell A \mid \Diamond^{\ell+1} \text{[hd][\varphi]} \} \]
We have to show
\[ M(g, s, t) : \{ \text{CoList}^\ell A \mid \Diamond^{k+\ell+1} \text{[hd][\varphi]} \} \]
Using
\[ \Diamond^{k+1} \text{[nil]} \Leftrightarrow \text{[nil]} \lor \Diamond^{k} \text{[nil]} \]
we apply the (\lor-\text{E}) rule on the refinement type of s. This leads to two subcases.

**(Sub)** Case of [nil].
We have
\[ \text{unfold}(s) : \{ 1 + A \times \text{CoList}^\ell A \mid [\text{in}_0] \top \} \]
Since \[ [\ell+1] \leq [k+1+\ell] \], the result then follows by applying the (INJ_0-\text{E}) rule.
{(Sub)Case of} \(\bigcirc \diamond k[\text{nil}]\).

We have

\[
\text{unfold}(s) : \{ 1 + A \times \triangleright \text{CoList}^6 A \mid [\text{in}_1][\pi_1][\text{next}]\bigcirc k[\text{nil}] \}.
\]

Using the (INJ1-E) rule we are left with showing

\[
\text{Cons}^6 x (g \odot xs \odot (\text{next} t)) : \{ \text{CoList}^6 A \mid \diamond k[\text{nil}] \}
\]

where

\[
\begin{align*}
x & := \pi_0(y) : A \quad x_s := \pi_1(y) : \triangleright \{ \text{CoList}^6 A \mid \diamond k[\text{nil}] \} \\
\end{align*}
\]

assuming

\[
\begin{align*}
y & : \{ A \times \triangleright \text{CoList}^6 A \mid [\pi_1][\text{next}][\diamond k[\text{nil}] \}
\end{align*}
\]

We have

\[
g \odot x_s \odot (\text{next} t) : \triangleright \{ \text{CoList}^6 A \mid \diamond k \ell[\text{hd}][\varphi] \}
\]

It follows that

\[
\begin{align*}
\text{Cons}^6 x (g \odot x_s \odot (\text{next} t)) & : \{ \text{CoList}^6 A \mid \bigcirc \diamond k \ell[\text{hd}][\varphi] \} \\
\end{align*}
\]

and we are done since

\[
\bigcirc \diamond k \ell[\text{hd}][\varphi] \Rightarrow \diamond k + 1 + 1 \ell[\text{hd}][\varphi]
\]

\(\square\)

E.7 Resumptions

This example is adapted from [48]. Fix a constant type \(0\) and a finite base type \(I\). Let

\[
\begin{align*}
\text{Res} A & := \blacksquare \text{Res}^6 A \\
\text{Res}^6 A & := \text{Fix}(X).A + (0 \times \triangleright X)^I
\end{align*}
\]

and

\[
\begin{align*}
\text{Ret}^6 & := \lambda a. \text{fold}(\text{in}_0 a) : A \rightarrow \text{Res}^6 A \\
\text{Cont}^6 & := \lambda k. \text{fold}(\text{in}_1 k) : (0 \times \triangleright \text{Res}^6 A)^I \rightarrow \text{Res}^6 A
\end{align*}
\]

Example E.40 (A Scheduler on Resumptions).

\[
\begin{align*}
\text{sched} : \quad \text{Res} A & \rightarrow \text{Res} A \rightarrow \text{Res} A \\
& := \lambda p. \lambda q. \text{box}(\text{sched}^6 (\text{unbox} p) (\text{unbox} q))
\end{align*}
\]

\[
\begin{align*}
\text{sched}^6 : \quad \text{Res}^6 A & \rightarrow \text{Res}^6 A \rightarrow \text{Res}^6 A \\
& := \text{fix}(g).\lambda p. \lambda q. \text{case} p \text{ of} \\
& \mid \text{Ret}^6 a \mapsto \text{Ret}^6 a \\
& \mid \text{Cont}^6 k \mapsto \\
& \quad \text{let} \ h = \lambda i. \text{let} \ \langle o, t \rangle = ki \\
& \quad \text{in} \ \langle o, g \odot (\text{next} q) \odot t \rangle \\
& \quad \text{in} \ \text{Cont}^6 h
\end{align*}
\]
Here, Ret\(^\$\)(a) represents a computation which returns the value \(a : A\), while Cont\(^\$\)(f,k) (with \((f,k) : I \rightarrow (0 \times ▶ Ret\(^\$\)A)) represents a computation which on input \(i : I\) outputs \(f i : 0\) and continues with the computation \(k i : ▶ Ret\(^\$\)A\).

Provided with resumptions \(p,q : Ret\(^\$\)A\), the scheduler \((sched\(^\$\) p q)\), adapted from \[38\], first evaluates \(p\). If \(p\) returns, then the whole computation returns, with the same value. Otherwise, \(p\) evaluates to say Cont\(^\$\)(f,k). Then \((sched\(^\$\) p q)\) produces a computation which on input \(i : I\) outputs \(f i\) and continues with the computation \((sched\(^\$\) q (k i))\), thus switching arguments.

**Example E.41 (Formulae on Ret\(^\$\)A).** Assuming \(\psi : A, \varphi : Ret\(^\$\)A, \vartheta : 0\) and \(i \in I\),

\[
\begin{align*}
[Ret] & := [fold][in_0]T : Ret\(^\$\)A \\
[Cont] & := [fold][in_1]T : Ret\(^\$\)A \\
[now]\psi & := [fold][in_0]\psi : Ret\(^\$\)A \\
[out_1]\vartheta & := [fold][in_1]([i] [\rightarrow [\pi_0]\vartheta]) : Ret\(^\$\)A \\
[\land out]\vartheta & := \land_{i \in I}[out_i]\vartheta : Ret\(^\$\)A \\
[\lor out]\vartheta & := \lor_{i \in I}[out_i]\vartheta : Ret\(^\$\)A
\end{align*}
\]

\(\bigcirc_1 \varphi := [fold][in_1]([i] [\rightarrow [\pi_1][next]\varphi]) : Ret\(^\$\)A\)

\(\emptyset \varphi := \land_{i \in I} \bigcirc_1 \varphi : Ret\(^\$\)A\)

\(\bullet \varphi := \lor_{i \in I} \bigcirc_1 \varphi : Ret\(^\$\)A\)

\(\exists \Box \varphi := \nu \alpha. \varphi \land \Box \alpha : Ret\(^\$\)A\)

\(\forall \Box \varphi := \nu \alpha. \varphi \land \Box \alpha : Ret\(^\$\)A\)

\(\exists \Diamond \varphi := \mu \alpha. \varphi \lor \Diamond \alpha : Ret\(^\$\)A\)

\(\forall \Diamond \varphi := \mu \alpha. \varphi \lor \Diamond \alpha : Ret\(^\$\)A\)

We moreover let

\[
\begin{align*}
\forall \Box^\top \psi & := \nu^\top \alpha. \psi \land \Box \alpha : Ret\(^\$\)A \\
\exists \Box^\top \psi & := \nu^\top \alpha. \psi \land \Box \alpha : Ret\(^\$\)A \\
\forall \Diamond^\top \psi & := \mu^\top \alpha. \psi \lor \Diamond \alpha : Ret\(^\$\)A \\
\exists \Diamond^\top \psi & := \mu^\top \alpha. \psi \lor \Diamond \alpha : Ret\(^\$\)A
\end{align*}
\]

The formula \(\exists \Diamond \varphi\) holds on a resumption if there is a finite sequence of inputs which leads to a resumption satisfying \(\varphi\), while \(\forall \Diamond \varphi\) holds on a resumption if \(\varphi\) holds at some point for any infinite sequence of inputs (this relies on Weak König Lemma). Moreover, \(\exists \Box \varphi\) expresses that there is an infinite sequence of inputs in which the resumption never returns and along which \(\varphi\) always holds, while \(\forall \Box \varphi\) expresses that for all infinite sequence of inputs, the resumption never returns and \(\varphi\) always holds. For instance, the composite formula \(\exists \Box \Diamond \Box [Ret]\) says that there is an infinite sequence of inputs along which (1) the resumption does not return and (2), at any point, there is a finite sequence of inputs which leads to a return.
Example E.42. Let $\psi : A$ be a safe and smooth formula and let $\varphi \in \{[\operatorname{Ret}], [\operatorname{now}] \psi\}$.

We have

\[
\begin{align*}
\text{sched} : & \{\text{Res } A \mid \box\exists \varphi\} \rightarrow \{\text{Res } A \mid \box\exists \varphi\} \\
\text{sched} : & \{\text{Res } A \mid \box\forall \varphi\} \rightarrow \{\text{Res } A \mid \box\forall \varphi\}
\end{align*}
\]

\[
\begin{align*}
\text{sched}^\#: & \forall k \cdot \forall \ell \cdot \{\{\text{Res}^\# A \mid \box^k \varphi\} \rightarrow \{\text{Res}^\# A \mid \box^k \varphi\} \rightarrow \{\text{Res}^\# A \mid \box^{k+\ell} \varphi\}\} \\
\text{sched}^\#: & \forall k \cdot \forall \ell \cdot \{\{\text{Res}^\# A \mid \forall^k \varphi\} \rightarrow \{\text{Res}^\# A \mid \forall^k \varphi\} \rightarrow \{\text{Res}^\# A \mid \forall^{k+\ell} \varphi\}\}
\end{align*}
\]

Proof. Let $\diamond \in \{\exists, \forall\}$ and

\[
T(k, \ell) := \{\text{Res}^\# A \mid \diamond^k \varphi\} \rightarrow \{\text{Res}^\# A \mid \diamond^\ell \varphi\} \rightarrow \{\text{Res}^\# A \mid \diamond^{k+\ell} \varphi\}
\]

and assume

\[
g : \forall k \cdot \forall \ell \cdot T(k, \ell)
\]

Let

\[
M(g, p, q) := \text{case } p \text{ of}
\]

\[
| \text{Ret}^\# a \mapsto \text{Ret}^\# a \\
| \text{Cont}^\# k \mapsto
\]

\[
\begin{align*}
& \begin{align*}
\text{let } h = \lambda i. \text{ let } (a, t) = ki \\
& \text{ in } (a, g \odot (\text{next } q) \odot t)
\end{align*}
\end{align*}
\]

\[
\text{in } \text{ Cont}^\# h
\]

We show

\[
\lambda p. \lambda q. M(g, p, q) : \forall k \cdot \forall \ell \cdot T(k, \ell)
\]

We apply the ($\forall$-CI) rule on $\forall k$. In the case of $\forall \ell \cdot T(0, \ell)$, we get the result using the (ExF) rule since

\[
\diamond^0 \varphi \leftrightarrow \bot
\]

As for $\forall \ell \cdot T(k+1, \ell)$, we apply the ($\forall$-I) rule on $\forall \ell$. We have to show

\[
M(g, p, q) : \{\text{Res}^\# A \mid \diamond^{k+\ell+1} \varphi\}
\]

assuming

\[
p : \{\text{Res}^\# A \mid \diamond^{k+1} \varphi\} \\
q : \{\text{Res}^\# A \mid \diamond^\ell \varphi\}
\]

Using

\[
\exists \diamond^{k+1} \varphi \leftrightarrow \varphi \vee \exists \diamond^k \varphi \\
\forall \diamond^{k+1} \varphi \leftrightarrow \varphi \vee \forall \diamond^k \varphi
\]

we reason by cases on the refinement type of $p$.

Case of $[\operatorname{Ret}]$.

We have

\[
\text{unfold } p : \{A + (0 \times \box\text{Res}^\# A)^2 \mid \text{[in0]} \top\}
\]

We apply the (IN.in0-E) rule on $p$ and we are done since

\[
\text{Ret}^\# a : \{\text{Res}^\# A \mid [\text{Ret}]\}
\]

assuming

\[
a : A
\]
Case of $\text{now} \psi$.

We have

$$\text{unfold } p : \{ A + (0 \times \textbf{Res}^\psi A)_1 | \text{in}_0 \psi \}$$

We apply the (IN$_0$-E) rule on $p$ and we are done since

$$\text{Ret}^\psi a : \{ \text{Res}^\psi A | \text{now} \psi \}$$

assuming

$$a : \{ A | \psi \}$$

Case of $\forall \exists \psi \varphi$.

We apply the ($\forall$-E) rule on the refinement type of $p$. So let $i \in I$ and assume

$$p : \{ \text{Res}^\psi A | \bigcirc_i \exists \psi \varphi \}$$

We have

$$\text{unfold } p : \{ A + (0 \times \textbf{Res}^\psi A)_1 | [i] \rightarrow [\pi_1] [\text{next}] \exists \psi \varphi \}$$

We apply the (IN$_1$-E) rule on the refinement type of $p$. Let

$$N(g, k, q) := \text{let } h = \lambda i. \text{let } \langle o, t \rangle = ki \text{ in } \langle o, g \odot (\text{next } q) \odot t \rangle$$

in $\text{Cont}^\psi h$

We show

$$N(g, k, q) : \{ \text{Res}^\psi A | \bigcirc_1 \exists \psi \varphi \}$$

assuming

$$k : \{(0 \times \textbf{Res}^\psi A)_1 | [i] \rightarrow [\pi_1] [\text{next}] \exists \psi \varphi \}$$

Assuming

$$i : \{ I | [i] \}$$

we thus have

$$ki : \{ 0 \times \textbf{Res}^\psi A | [\pi_1] [\text{next}] \exists \psi \varphi \}$$

It follows that

$$\langle \pi_0(ki), g \odot (\text{next } q) \odot (\pi_1(ki)) \rangle : \{ 0 \times \textbf{Res}^\psi A | [\pi_1] [\text{next}] \exists \psi \varphi \}$$

and thus

$$\lambda i. \langle \pi_0(ki), g \odot (\text{next } q) \odot (\pi_1(ki)) \rangle : \{(0 \times \textbf{Res}^\psi A)_1 | [i] \rightarrow [\pi_1] [\text{next}] \exists \psi \varphi \}$$

Now we are done since

$$\bigcirc_1 \exists \psi \varphi = \text{fold} [\text{in}_1 \} \{ [i] \rightarrow [\pi_1] [\text{next}] \exists \psi \varphi \}$$

and

$$\text{Cont}^\psi = \lambda h. \text{fold} [\text{in}_1 \} h$$
Case of $\forall \Diamond \Diamond^k \varphi$.

Using

$$\forall \Diamond^{k+\ell} \varphi \iff \varphi \vee \Diamond \forall \Diamond^{k+\ell} \varphi$$

for each $i \in I$ we show

$$M(g, p, q) : \{\text{Res}^A | \Diamond \forall \Diamond^{k+\ell} \varphi\}$$

So let $i \in I$. Since

$$p : \{\text{Res}^A | \Diamond \exists \Diamond^k \varphi\}$$

We have

$$\text{unfold } p : \{A + (0 \times \triangleleft \text{Res}^A)^1 \mid \{\text{in} \}_1 \{[i] \mid [i_1][\text{next} \exists \Diamond^k \varphi]\}$$

and we conclude similarly as in the case of $\Diamond \exists \Diamond^k \varphi$. $\square$

Example E.43. Let $\vartheta : 0$ be a safe and smooth formula. Furthermore, let $\Box \in \{\forall, \exists \Diamond\}, \Diamond \in \{\forall \Diamond, \exists \Diamond^2\}$ and $[\text{out}] \in \{[\text{in}], [\text{out}], [\text{out}]\}$. We have

$sched : \{\text{Res} A \mid [\text{box}]\Diamond \Diamond^k [\text{out}] \vartheta\} \rightarrow \{\text{Res} A \mid [\text{box}]\Diamond \Diamond^k [\text{out}] \vartheta\} \rightarrow \{\text{Res} A \mid [\text{box}]\Diamond \Diamond^k [\text{out}] \vartheta\}$

Proof. We show that we can give the following refinement type to sched$^k$:

$$\forall k : \forall \ell_0 \forall \ell_1 \forall \ell_0 \forall \ell_1 : \{\text{Res}^A \mid \Diamond^k \Diamond^\ell_0 [\text{out}] \vartheta\} \rightarrow \{\text{Res}^A \mid \Diamond^k \Diamond^\ell_1 [\text{out}] \vartheta\} \rightarrow \{\text{Res}^A \mid \Diamond^k \Diamond^\ell_0+\ell_1 [\text{out}] \vartheta\}$$

Let $T(k, \ell_0, \ell_1)$ be the type

$$\forall k : \forall \ell_0 \forall \ell_1 : \forall \ell_0 \forall \ell_1 : T(k, \ell_0, \ell_1)$$

and assume

$$g : \rightarrow \forall k : \forall \ell_0 \forall \ell_1 : T(k, \ell_0, \ell_1)$$

Let

$$M(g, p, q) := \text{case } p \text{ of }$$

$$\text{Ret}^A a \mapsto \text{Ret}^A a$$

$$\text{Cont}^A k \mapsto$$

$$\begin{array}{l}
\text{let } h = \lambda. \text{let } (a, t) = k i \\
\text{in } (a, g \odot (\text{next } q) \odot t) \\
\text{in } \text{Cont}^A h
\end{array}$$

We show

$$\lambda p. \lambda q. M(g, p, q) : \forall k : \forall \ell_0 \forall \ell_1 : T(k, \ell_0, \ell_1)$$

We apply the $(\forall$-CI) rule on $\forall k$. The case of $\forall \ell_0 \forall \ell_1 : T(0, \ell_0, \ell_1)$ is trivial since

$$\Box^0 \Diamond^0 [\text{out}] \vartheta \iff \top$$

As for $\forall \ell_0 \forall \ell_1 : T(k+1, \ell_0, \ell_1)$, we apply the $(\forall$-CI) rule, this time on $\forall \ell_0$. In the case of $\forall \ell_1 : T(k+1, 0, \ell_1)$, since $\Box^{k+1} \Diamond^0 [\text{out}] \vartheta$ is of the form

$$\Diamond^0 [\text{out}] \vartheta \land \psi$$
while

\[ \Diamond^\theta[\text{out}] \vartheta \leftrightarrow \bot \]

we can conclude using the (ExF) rule. It remains to deal with the case of \( \forall \ell_1 \cdot T(k+1,\ell_0+1,\ell_1) \). We apply the (\( \forall \)-I) rule on \( \forall \ell_1 \). We show

\[ M(g,p,q) : \{ \text{Res}^\vartheta A \mid \Box^{k+1} \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \} \]

assuming

\[ p : \{ \text{Res}^\vartheta A \mid \Box^{k+1} \Diamond^{\ell_0+1} \text{out}]\vartheta \} \]

\[ q : \{ \text{Res}^\vartheta A \mid \Box^{k+1} \Diamond^{\ell_1} \text{out}]\vartheta \} \]

We will apply the (Inj\(_1\)-E) rule on (unfold \( p \)) and show

\[ N(g,k,q) : \{ \text{Res}^\vartheta A \mid \Box^{k+1} \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \} \]

where

\[ N(g,k,q) := \text{let } h = \lambda \ell_1. \text{let } \langle o,t \rangle = k_i \text{ in } \langle o,g \circ \text{next} q \circ t \rangle \]

in \text{Cont}^\vartheta h

and under suitable assumption on the refinement type of \( k \). We have

\[ \forall \Box^{k+1} \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \leftrightarrow \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \land \forall \forall \Box^{k+1} \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \land \exists \exists \Box^{k+1} \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \]

and we consider each conjunct separately.

**Cases of** \( \Diamond^{\ell_0+\ell_1+1} \text{out}]\vartheta \).

We have

\[ p : \{ \text{Res}^\vartheta A \mid \Diamond^{\ell_0+1} \text{out}]\vartheta \} \]

Using

\[ \exists \Diamond^{\ell_0+1} \text{out}]\vartheta \leftrightarrow [\text{out}]\vartheta \lor \Diamond \exists \Diamond^{\ell_0} \text{out}]\vartheta \]

\[ \forall \Diamond^{\ell_0+1} \text{out}]\vartheta \leftrightarrow [\text{out}]\vartheta \lor \Diamond \forall \Diamond^{\ell_0} \text{out}]\vartheta \]

we reason by cases on the refinement type of \( p \).

**(Sub)Cases of** \([\text{out}]\vartheta \).

We show

\[ N(g,k,q) : \{ \text{Res}^\vartheta A \mid [\text{out}]\vartheta \} \]

We handle the cases of \([\lor \text{out}]\) and \([\land \text{out}]\) separately.

**(SubSub)Case of** \([\lor \text{out}]\).

We apply the (\( \lor \)-E) rule on the refinement type of \( p \). So let \( i \in I \) and assume

\[ p : \{ \text{Res}^\vartheta A \mid [\text{out}_i]\vartheta \} \]

This amounts to

\[ k : \{ (0 \times \text{Res}^\vartheta A)^\top \mid [i] \rightarrow [\pi_0]\vartheta \} \]

Hence assuming

\[ i : \{ A \mid [i] \} \]
we have

\[
(\pi_0(k_i), g \circledast (\text{next } q) \circledast (\pi_1(k_i))) : \{0 \times \triangleright \text{Res}^g A \mid [\pi_0]q\}
\]

It follows that

\[
\lambda i. (\pi_0(k_i), g \circledast (\text{next } q) \circledast (\pi_1(k_i))) : \{(0 \times \triangleright \text{Res}^g A)^I \mid [i] \rightarrow [\pi_0]q\}
\]

and we are done since

\[
\text{Cont}^g = \lambda h. \text{fold}(\text{in}_1 h)
\]

(SubSub)Case of $[\land \text{out}]$.

For each $i \in I$ we have to show

\[
N(g, k, q) : \{\text{Res}^g A \mid [\text{out}_i]q\}
\]

So let $i \in I$. Since

\[
p : \{\text{Res}^g A \mid [\text{out}_i]q\}
\]

we have

\[
k : \{(0 \times \triangleright \text{Res}^g A)^I \mid [i] \rightarrow [\pi_0]q\}
\]

and we conclude similarly as in the case of $[\lor \text{out}]$.

(Sub)Case of $\lor \exists^\ell_0 [\text{out}]q$.

We show

\[
N(g, k, q) : \{\text{Res}^g A \mid \lor \exists^\ell_0 [\text{out}]q\}
\]

We apply the ($\lor$-E) rule on the refinement type of $p$. So let $i \in I$ and assume

\[
p : \{\text{Res}^g A \mid \lor_1 \exists^\ell_0 [\text{out}]q\}
\]

This amounts to

\[
k : \{(0 \times \triangleright \text{Res}^g A)^I \mid [i] \rightarrow [\pi_1][\text{next}]\exists^\ell_0 [\text{out}]q\}
\]

Assuming

\[
i : \{I \mid [i]\}
\]

we thus have

\[
ki : \{0 \times \triangleright \text{Res}^g A \mid [\pi_1][\text{next}]\exists^\ell_0 [\text{out}]q\}
\]

since (by subtyping) $g$ has type

\[
\triangleright \left(\{\text{Res}^g A \mid \square \exists^\ell_0 [\text{out}]q\} \rightarrow \{\text{Res}^g A \mid \square \exists^\ell_1 [\text{out}]q\}\right) \rightarrow \{\text{Res}^g A \mid \square \exists^\ell_0 + \ell_1 [\text{out}]q\}
\]

and since, according to Table 2

\[
\square \exists \theta \iff \theta
\]
it follows that 
\[
\langle \pi_0(k_i), g \circ (\text{next } q) \circ (\pi_1(k_i)) \rangle : \{ 0 \times \triangleright_{\text{Res}^\delta} A \mid [\pi_1][\text{next}] \exists \lambda^i \cdot \phi_i \cdot \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
We thus get 
\[
\lambda i. \langle \pi_0(k_i), g \circ (\text{next } q) \circ (\pi_1(k_i)) \rangle : \{ (0 \times \triangleright_{\text{Res}^\delta} A)^1 \mid [1] \rightarrow [\pi_1][\text{next}] \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
Now we are done since 
\[
\bigcirc_1 \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta = [\text{fold}][\text{in}_1] (\{1\} \rightarrow [\pi_1][\text{next}] \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta)
\]
and 
\[
\text{Cont}^\delta = \lambda h. \text{fold}(\text{in}_1 \ h)
\]
(Sub)Case of \( \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \).
We show 
\[
N(g, k, q) : \{ \text{Res}^\delta A \mid \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
Hence, for each \( i \in I \) we have to show 
\[
N(g, k, q) : \{ \text{Res}^\delta A \mid \bigcirc_1 \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
So let \( i \in I \). Since 
\[
p : \{ \text{Res}^\delta A \mid \bigcirc_1 \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
we have 
\[
k : \{ (0 \times \triangleright_{\text{Res}^\delta} A)^1 \mid [1] \rightarrow [\pi_1][\text{next}] \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
and we conclude similarly as in the case of \( \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \).
Case of \( \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \).
For each \( i \in I \) we have to show 
\[
N(g, k, q) : \{ \text{Res}^\delta A \mid \bigcirc_1 \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
So let \( i \in I \). Since 
\[
p : \{ \text{Res}^\delta A \mid \bigcirc_1 \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
we have 
\[
k : \{ (0 \times \triangleright_{\text{Res}^\delta} A)^1 \mid [1] \rightarrow [\pi_1][\text{next}] \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
Assuming 
\[
i : \{ I \mid [1] \}
\]
we thus have 
\[
ki : \{ 0 \times \triangleright_{\text{Res}^\delta} A \mid [\pi_1][\text{next}] \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
and it follows that 
\[
\lambda i. \langle \pi_0(k_i), g \circ (\text{next } q) \circ (\pi_1(k_i)) \rangle : \{ (0 \times \triangleright_{\text{Res}^\delta} A)^1 \mid [1] \rightarrow [\pi_1][\text{next}] \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta \}
\]
Now we are done since 
\[
\bigcirc_1 \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta = [\text{fold}][\text{in}_1] (\{1\} \rightarrow [\pi_1][\text{next}] \forall \forall \exists \lambda^i \cdot \phi_i \cdot [\text{out}] \theta)
\]
and 
\[
\text{Cont}^\delta = \lambda h. \text{fold}(\text{in}_1 \ h)
\]
Case of $\forall \exists^k \diamond^a \ell_0 + \ell_1 \mid \text{out} \diamond^a$.

We have to show

$$\mathcal{N}(g, k, q) : \{\text{Res}^g A \mid \forall \exists^k \diamond^a \ell_0 + \ell_1 \mid \text{out} \diamond^a\}$$

We apply the ($\forall$-E) rule on the refinement type of $p$. So let $i \in I$ and assume

$$p : \{\text{Res}^g A \mid \bigcirc_i \forall \exists^k \diamond^a \ell_0 + \ell_1 \mid \text{out} \diamond^a\}$$

We have

$$k : \{(0 \times \Box \text{Res}^g A)^I \mid [i] \mapsto [\pi_i][\text{next}] \exists^k \diamond^a \ell_0 + \ell_1 \mid \text{out} \diamond^a\}$$

and we conclude similarly as in the case of $\forall \exists^k \diamond^a \ell_0 + \ell_1 \mid \text{out} \diamond^a$. 

Example E.44. Let $\Box \in \{\forall, \exists\}$ and $\diamond \in \{\forall, \exists\}$. We have

sched : $\{\text{Res} A \mid [\text{box}]\Box \diamond \text{[Ret]}\} \rightarrow \{\text{Res} A \mid [\text{box}]\Box \diamond \text{[Ret]}\} ightarrow \{\text{Res} A \mid [\text{box}]\Box \diamond \text{[Ret]}\}$

Proof. We show that we can give the following refinement type to sched$^g$:

$$\forall k : \forall \ell_0 : \forall \ell_1 : \{\text{Res}^g A \mid \Box k \diamond^a \text{[Ret]}\} \rightarrow \{\text{Res}^g A \mid \Box k \diamond^a \text{[Ret]}\} \rightarrow \{\text{Res}^g A \mid \Box k \diamond^a \ell_1 \mid \text{[Ret]}\}$$

Let $T(k, \ell_0, \ell_1)$ be the type

$$\{\text{Res}^g A \mid \Box k \diamond^a \text{[Ret]}\} \rightarrow \{\text{Res}^g A \mid \Box k \diamond^a \text{[Ret]}\} \rightarrow \{\text{Res}^g A \mid \Box k \diamond^a + \ell_1 \mid \text{[Ret]}\}$$

and assume

$$g : \bigtriangledown \forall k : \forall \ell_0 : \forall \ell_1 : T(k, \ell_0, \ell_1)$$

Let

$$M(g, p, q) := \text{case } p \text{ of }$$

$$\text{Ret}^g a \mapsto \text{Ret}^g a$$

$$\text{Cont}^g k \mapsto$$

let $h = \lambda i. \ let (o, t) = ki$

in $\langle o, g \oplus (\text{next } q) \oplus t \rangle$

$$\in \text{Cont}^g h$$

We show

$$\lambda p. \lambda q. M(g, p, q) : \forall k : \forall \ell_0 : \forall \ell_1 : T(k, \ell_0, \ell_1)$$

We apply the ($\forall$-CI) rule on $\forall k$. The case of $\forall \ell_0 : \forall \ell_1 : T(0, \ell_0, \ell_1)$ is trivial since

$$\Box^0 \diamond^a [\text{Ret}] \Rightarrow \top$$

As for $\forall \ell_0 : \forall \ell_1 : T(k+1, \ell_0, \ell_1)$, we apply the ($\forall$-CI) rule, this time on $\forall \ell_0$. In the case of $\forall \ell_1 : T(k+1, 0, \ell_1)$, since $\Box^k \diamond^a [\text{Ret}]$ is of the form

$$\diamond^0 [\text{Ret}] \land \psi$$

while

$$\diamond^0 [\text{Ret}] \Leftrightarrow \bot$$

we can conclude using the (ExF) rule. It remains to deal with the case of \( \forall \ell_1 : T(k+1, \ell_0+1, \ell_1) \). We apply the (\( \forall \)-I) rule on \( \forall \ell_1 \). We show
\[
M(g, p, q) : \{ \text{Res}^g A \mid \Box^{k+1} \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \}
\]
assuming
\[
p : \{ \text{Res}^g A \mid \Box^{k+1} \Diamond \ell_1 + 1 \text{[Ret]} \}
\]
\[
q : \{ \text{Res}^g A \mid \Box^{k+1} \Diamond \ell_1 \text{[Ret]} \}
\]
We have
\[
\forall \Box^{k+1} \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \iff \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \land \forall \Box^k \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]}
\]
\[
\exists \Box^{k+1} \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \iff \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \land \forall \exists \Box^k \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]}
\]
and we consider each conjunct separately.

**Cases of** \( \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \).
We have
\[
p : \{ \text{Res}^g A \mid \Diamond \ell_0 + 1 \text{[out]} \}
\]
Using
\[
\exists \Diamond \ell_0 + 1 \text{[Ret]} \iff \exists \Diamond \ell_0 \text{[Ret]} \lor \forall \exists \Diamond \ell_0 \text{[Ret]}
\]
\[
\forall \Diamond \ell_0 + 1 \text{[Ret]} \iff \forall \exists \Diamond \ell_0 \text{[Ret]} \lor \forall \forall \Diamond \ell_0 \text{[Ret]}
\]
we reason by cases on the refinement type of \( p \). In the case of [Ret], apply the (InJ\(0\)-E) rule on (unfold \( p \)), and we conclude similarly as in Ex. [E.42] In the other cases, we apply the (InJ\(1\)-E) rule on (unfold \( p \)) and show
\[
N(g, k, q) : \{ \text{Res}^g A \mid \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \}
\]
where
\[
N(g, k, q) := \text{let } h = \lambda i. \text{ let } \langle o, t \rangle = ki \text{ in } \langle o, g \odot (\text{next } q) \odot t \rangle \text{ in Cont}^g h
\]
and under suitable assumption on the refinement type of \( k \). We can then conclude similarly as in Ex. [E.43]

**Cases of** \( \Box^{k+1} \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \).
We apply the (InJ\(1\)-E) rule on (unfold \( p \)) and show
\[
N(g, k, q) : \{ \text{Res}^g A \mid \Box^{k+1} \Diamond \ell_0 + \ell_1 + 1 \text{[Ret]} \}
\]
where
\[
N(g, k, q) := \text{let } h = \lambda i. \text{ let } \langle o, t \rangle = ki \text{ in } \langle o, g \odot (\text{next } q) \odot t \rangle \text{ in Cont}^g h
\]
and under suitable assumption on the refinement type of \( k \). We can then conclude similarly as in Ex. [E.43]
E.8 Breadth-First Tree Traversal

Infinite Binary Trees The guarded recursive type of binary trees is

\[
\text{Tree}^g A := \text{Fix}(X).A \times (\llbracket X \rrbracket \times \llbracket X \rrbracket)
\]
\[
\text{Tree} A := \llbracket \text{Tree}^g A \rrbracket
\]

The usual guarded constructors and destructors on \(\text{Tree}^g A\) are represented as

\[
\begin{align*}
\text{Node}^g & := \lambda \ell. \lambda r. \text{fold}(v, \llbracket \ell, r \rrbracket) : A \to \text{Tree}^g A \to \text{Tree}^g A \\
\text{label}^g & := \lambda t. \pi_0(\text{unfold } t) : \text{Tree}^g A \to A \\
\text{son}_\ell^g & := \lambda t. \pi_0(\text{π}_1(\text{unbox } t)) : \text{Tree}^g A \to \llbracket \text{Tree}^g A \rrbracket \\
\text{son}_r^g & := \lambda t. \pi_1(\text{unbox } t) : \text{Tree}^g A \to \llbracket \text{Tree}^g A \rrbracket
\end{align*}
\]

Their coinductive (for \(A\) a constant type) variants are

\[
\begin{align*}
\text{Node} & := \lambda \ell. \lambda r. \\
&\text{box}_v(\text{Node}^g (\text{next } (\text{unbox } \ell)) (\text{next } (\text{unbox } \ell))) : A \to \text{Tree} A \to \text{Tree} A \to \text{Tree} A \\
\text{label} & := \lambda t. \text{label}^g (\text{unbox } t) : \text{Tree} A \to A \\
\text{son}_\ell & := \lambda t. \text{son}_\ell^g (\text{unbox } t) : \text{Tree} A \to \llbracket \text{Tree}^g A \rrbracket \\
\text{son}_r & := \lambda t. \text{son}_r^g (\text{unbox } t) : \text{Tree} A \to \llbracket \text{Tree}^g A \rrbracket
\end{align*}
\]

\textbf{Example E.45 (Tree Formulae).} Assuming \(\varphi : \text{Tree}^g A\),

\[
\begin{align*}
\forall \Box \varphi & : \text{Tree}^g A \\
&:= \nu \alpha. \varphi \land (\llbracket \alpha \rrbracket \land \lozenge \alpha) \\
\exists \Diamond \varphi & : \text{Tree}^g A \\
&:= \mu \alpha. \varphi \lor (\llbracket \alpha \rrbracket \lor \Diamond \alpha)
\end{align*}
\]

\textbf{Example E.46.} Assuming \(\varphi : A\), we have

\[
\begin{align*}
\text{Node}^g & : \{A \mid \varphi\} \to \llbracket \text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket \to \llbracket \text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket \\
\text{label}^g & : \{\text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket \to \{A \mid \varphi\} \\
\text{son}_\ell^g & : \{\text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket \to \llbracket \text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket \\
\text{son}_r^g & : \{\text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket \to \llbracket \text{Tree}^g A \mid \forall \Box [\llbracket \llbracket \varphi \rrbracket] \rrbracket \rrbracket
\end{align*}
\]

\textbf{Breadth-First Traversal of Guarded Trees Using Forests}
Example E.47.

\[
\text{bft} : \text{Tree } A \rightarrow \text{CoList } A \\
:= \lambda t. \text{box } \iota (\text{bft } g (\text{unbox } t))
\]

\[
\text{bft}^g : \text{Tree}^g A \rightarrow \text{CoList}^g A \\
:= \lambda t. \text{bftaux}^g [t]^g
\]

\[
\text{bftaux}^g : \text{CoList}^g (\text{Tree}^g A) \rightarrow \text{CoList}^g A \\
:= \text{fix}(g). \lambda s. \text{case } s \text{ of} \\
\text{Nil}^g \mapsto \text{Nil}^g \\
\text{Cons}^g x xs \mapsto (\text{label}^g x) \mapsto g \circ (\text{append}^g) \circ xs \circ ([\text{son}^g x], \text{son}^g xs)^{g\bullet}
\]

where

\[
[]^{g\bullet} := \text{next}([\emptyset]^g) \\
[y_0, y_1, \ldots, y_n]^{g\bullet} := \text{next} (\text{Cons}^g) \circ y_0 \circ \text{next} [y_1, \ldots, y_n]^{g\bullet}
\]

Example E.48.

\[
\text{bft}^g : \text{Tree}^g A \rightarrow \{\text{CoList}^g A | \neg \text{nil}\}
\]

\[
\text{bftaux}^g : \{\text{CoList}^g (\text{Tree}^g A) | \neg \text{nil}\} \rightarrow \{\text{CoList}^g A | \neg \text{nil}\}
\]

Example E.49.

\[
\text{bft}^g : \text{Tree } A \rightarrow \{\text{CoList}^g A | \text{inf}\}
\]

\[
\text{bftaux}^g : \{\text{CoList}^g (\text{Tree} A) | \neg \text{nil}\} \rightarrow \{\text{CoList}^g A | \text{inf}\}
\]

Example E.50. Assuming \( \varphi : A \),

\[
\text{bft}^g : \{\text{Tree}^g A | \forall \Diamond [\text{bl}] \varphi\} \rightarrow \{\text{CoList}^g A | \Box [\text{hd}] \varphi\}
\]

Proof. Thanks to Ex. E.30 and Ex. E.31 we can reduce to showing

\[
\text{bftaux}^g : \{\text{CoList}^g (\text{Tree}^g A) | \neg \text{nil} \land \Box^\Diamond [\text{hd}] \forall \Diamond [\text{bl}] \varphi\} \rightarrow \{\text{CoList}^g A | \Box [\text{hd}] \varphi\}
\]

Let

\[
T := \{\text{CoList}^g (\text{Tree}^g A) | \neg \text{nil} \land \Box^\Diamond [\text{hd}] \forall \Diamond [\text{bl}] \varphi\} \rightarrow \{\text{CoList}^g A | \Box [\text{hd}] \varphi\}
\]

and assume

\[
g : \triangleright T \\
s : \{\text{CoList}^g (\text{Tree}^g A) | \neg \text{nil} \land \Box^\Diamond [\text{hd}] \forall \Diamond [\text{bl}] \varphi\}
\]

Note that we have, at type CoList^g(\text{Tree}^g A),

\[
\neg \text{nil} \land \Box^\Diamond [\text{hd}] \forall \Diamond [\text{bl}] \varphi \iff \neg \text{nil} \land (\neg \text{nil} \lor (\text{hd} \forall \Diamond [\text{bl}] \varphi \land \Box^\Diamond^\Diamond [\text{hd}] \forall \Diamond [\text{bl}] \varphi)) \\
\iff (\neg \text{nil} \land \neg \text{nil} \lor (\neg \text{nil} \land \text{hd} \forall \Diamond [\text{bl}] \varphi \land \Box^\Diamond^\Diamond [\text{hd}] \forall \Diamond [\text{bl}] \varphi))
\]
Since the modality $[\text{fold}]$ preserves $\wedge$ and $\perp$ (Table 2), we have

$([\neg\text{nil}] \wedge [\text{nil}]) \Rightarrow \perp$

We apply the $(\lor\text{-}E)$ rule on the refinement type of $s$. The branch of $[\neg\text{nil}] \wedge [\text{nil}]$ is dealt-with using the rule (ExF). It remains to handle the case of

$s: \{\text{CoList}^\delta(\text{Tree}^\delta A) \mid [\neg\text{nil}] \wedge [\text{nil}]\} + \neg \text{unf} \square [\text{hd}]\varphi \
\wedge \square \text{fin} \{\text{hd}\} \varphi\}$

Since the modalities $[\text{fold}]$ and $[\text{in}_1]$ preserve $\wedge$ we have

$\text{unfold}(s): \{1 + \text{Tree}^\delta A \times \text{CoList}^\delta(\text{Tree}^\delta A) \mid [\text{in}_1]\{([\pi_0]\varphi \wedge [\pi_1]\text{in} \text{fin}[\text{hd}]\neg \square[\text{lb}]\varphi\}\}$

Using the typing rule $(\text{In1-E})$ (Fig. 8) and Ex. [E.46] we are left with showing

$v: \gamma \circ (\text{next}(\text{append}) \circ \pi \circ [\ell, r]^{\text{sic}}) : \{\text{CoList}^\delta A \mid \neg [\text{hd}]\varphi\}$

where

$\pi \equiv \pi_1 \gamma$ : $\text{CoList}^\delta(\text{Tree}^\delta A) \mid \square \text{fin}[\text{hd}]\neg \square[\text{lb}]\varphi \}$

$v := \text{label}^\delta(\pi_0 \gamma) : \{A \mid \varphi\}$

$\ell := \text{son}^\delta(\pi_0 \gamma) : \{\text{Tree}^\delta A \mid \forall \square[\text{lb}]\varphi\}$

$r := \text{son}^\delta(\pi_0 \gamma) : \{\text{Tree}^\delta A \mid \forall \square[\text{lb}]\varphi\}$

assuming

$\gamma : \{\text{Tree}^\delta A \times \text{CoList}^\delta(\text{Tree}^\delta A) \mid [\pi_0]\varphi \wedge [\pi_1]\text{next} \text{fin}[\text{hd}]\neg \square[\text{lb}]\varphi\}$

It follows from Ex. [E.30] and Ex. [E.31] that

$[\ell, r]^{\text{sic}} : \text{CoList}^\delta(\text{Tree}^\delta A) \mid [\neg\text{nil}] \wedge [\text{nil}]\} + \neg \text{in} \text{fin}[\text{hd}]\neg \square[\text{lb}]\varphi\}$

Hence, by Ex. [E.32] and Ex. [E.33] we obtain

$\text{next}(\text{append}) \circ \pi \circ [\ell, r]^{\text{sic}} : \text{CoList}^\delta(\text{Tree}^\delta A) \mid [\neg\text{nil}] \wedge \text{in} \text{fin}[\text{hd}]\neg \square[\text{lb}]\varphi\}$

and the result follows.

**Martin Hofmann’s Algorithm** We follow the presentation of [10] with some slight changes in terminology and notation. Consider the non-strictly positive type

$\text{Rou}^\delta A := \text{Fix}(X). 1 + ((\text{CoList}^\delta A) \rightarrow A)$

so that

$\text{Rou}^\delta(\text{CoList}^\delta A) := \text{Fix}(X). 1 + ((\text{CoList}^\delta A) \rightarrow \text{CoList}^\delta A)$

The constructors of $\text{Rou}^\delta A$ are

$\text{Over}^\delta := \text{fold}(\text{in}_0()) : \text{Rou}^\delta A$

$\text{Cont}^\delta := \lambda f.\text{fold}(\text{in}_1 f) : ((\text{CoList}^\delta A) \rightarrow A) \rightarrow \text{Rou}^\delta A$
The following are two basic important functions on \( \text{Rou}^\# \):

\[
\text{unfold} : \quad \text{Rou}^\# A \rightarrow (\rightarrow \text{Rou}^\# A \rightarrow A) 
\quad := \lambda c. \text{case } c \text{ of } \\
| \text{Over}^\# \mapsto \lambda k. k (\text{next Over}^\#) \\
| \text{Cont} f \mapsto \lambda k. \text{next}(fk)
\]

\[
\text{extract} : \quad \text{Rou}^\# (\text{CoList}^\# A) \rightarrow \text{CoList}^\# A 
\quad := \text{fix}(g). \lambda c. \text{case } c \text{ of } \\
| \text{Over}^\# \mapsto \text{Nil}^\# \\
| \text{Cont} f \mapsto fg^\#
\]

where

\[
g^\# := \lambda x.g \odot x
\]

We then let

\[
\text{bft}^\# : \quad \text{Tree}^\# A \rightarrow \text{CoList}^\# A 
\quad := \lambda t. \text{extract } (\text{btaux } t \text{ Over}^\#)
\]

\[
\text{btaux} : \quad \text{Tree}^\# A \rightarrow \text{Rou}^\# (\text{CoList}^\# A) 
\rightarrow \text{Rou}^\# (\text{CoList}^\# A) 
\quad := \text{fix}(g). \lambda c. \text{case } c \text{ of } \\
\quad \text{Cont } \lambda k. (\text{label} t) \odot \text{unfold } c \odot (k \circ (g \odot (\text{son}^\# t)) \odot (g \odot (\text{son}^\# t)))
\]

**Example E.51** ((Non) Emptiness).

\[
\begin{align*}
\text{ov} &:= [\text{fold}][\text{in}0] \top : \text{Rou}^\# A \\
\text{ct} &:= [\text{fold}][\text{in}1] \top : \text{Rou}^\# A
\end{align*}
\]

**Example E.52.** Assuming \( \varphi : A \), we let

\[
[\text{Rou}]\varphi := \nu \alpha. [\text{fold}][\text{in}1](([\text{next}]\alpha \mapsto [\text{next}]\varphi) \mapsto \varphi) : \text{Rou}^\# A
\]

Then for \( \varphi : \text{CoList}^\# A \) we have

\[
\text{extract} : \{\text{Rou}^\# (\text{CoList}^\# A) | [\text{Rou}]\varphi \} \rightarrow \{\text{CoList}^\# A | \varphi\}
\]

**Proof.** Assume

\[
g : \rightarrow (\{\text{Rou}^\# (\text{CoList}^\# A) | [\text{Rou}]\varphi \} \rightarrow \{\text{CoList}^\# A | \varphi\})
\]

and let

\[
B := \text{CoList}^\# A
\]

Since

\[
[\text{Rou}]\varphi = \nu \alpha. [\text{fold}][\text{in}1](([\text{next}]\alpha \mapsto [\text{next}]\varphi) \mapsto \varphi)
\]

we have

\[
(\text{unfold } c) : \{1 + (\rightarrow \text{Rou}^\# B \rightarrow \rightarrow B) \rightarrow B | [\text{in}1](([\text{next}]\text{Rou} \varphi) \mapsto [\text{next}]\varphi) \mapsto \varphi\}
\]
We can thus apply the (INJ₁-E) rule, which leads us to showing
\[ f \ (\lambda x. g \odot x) : \{ B \mid \varphi \} \]
assuming
\[ f : \{ [\text{next}][\text{Rou}] B \rightarrow [\text{next}][\text{Rou}] \varphi \rightarrow [\text{next}][\varphi] \mid \rightarrow \varphi \} \]
that is
\[ f : ([\text{next}][\text{Rou}] B \mid [\text{Rou}] \varphi) \rightarrow [\text{next}][\text{Rou}] \{ B \mid \varphi \} \rightarrow \{ B \mid \varphi \} \]
But this is trivial, by assumption on the type of \( g \).

Example E.53. Assuming \( \varphi : A \) we have
\[
\text{unfold} : \text{Rou}^8 A \rightarrow ([\text{next}][\text{Rou}^8 A \rightarrow \{ A \mid \varphi \}) \rightarrow \{ A \mid \varphi \}
\]
Proof. Assume
\[
c : \text{Rou}^8 A
k : [\text{next}][\text{Rou}^8 A \rightarrow \{ A \mid \varphi \}
\]
f : ([\text{next}][\text{Rou}^8 A \mid [\text{Rou}] \varphi) \rightarrow [\text{Rou}^8 A \mid \{ A \mid \varphi \}) \rightarrow \{ A \mid \varphi \}
\]
Then we have
\[
k (\text{next Over}^8) : [\{ A \mid \varphi \}
\]
Moreover, by subtyping we have
\[
k : [\{ \text{Rou}^8 A \mid [\text{Rou}] \varphi \} \rightarrow \{ A \mid \varphi \}
\]
so that
\[
\text{next}(fk) : [\{ A \mid \varphi \}
\]

Example E.54. Assuming \( \varphi : A \) we have
\[
\text{bft}^8 : \{ \text{Tree}^8 A \mid \forall \exists [\text{lbl}] \varphi \} \rightarrow \{ \text{CoList}^8 A \mid [\text{hd}] \varphi \}
\]
Proof. It follows from the type of extract in Ex. E.52 that we are done if we show
\[
\text{bftaux} : \{ \text{Tree}^8 A \mid \forall \exists [\text{lbl}] \psi \} \rightarrow \text{Rou}^8 (\text{CoList}^8 A) \rightarrow \{ \text{Rou}^8 (\text{CoList}^8 A) \mid [\text{Rou}] [\text{hd}] \psi \}
\]
Let
\[
T := \{ \text{Tree}^8 A \mid \forall \exists [\text{lbl}] \psi \} \rightarrow \text{Rou}^8 (\text{CoList}^8 A) \rightarrow \{ \text{Rou}^8 (\text{CoList}^8 A) \mid [\text{Rou}] [\text{hd}] \psi \}
\]
and assume
\[
g : [\text{next}][\text{Rou}^8 A \rightarrow [\text{next}][\text{Rou}^8 (\text{CoList}^8 A) \rightarrow [\text{next}][\text{Rou}] [\text{hd}] \psi \}
\]
t : \{ \text{Tree}^8 A \mid \forall \exists [\text{lbl}] \varphi \}
\]
c : \text{Rou}^8 (\text{CoList}^8 A)
Using Ex. E.46 let

\[ \ell := \text{son}^g t : \rightarrow \{ \text{Tree}^g A \mid \forall [l] \varphi \} \]
\[ r := \text{son}^g t : \rightarrow \{ \text{Tree}^g A \mid \forall [l] \varphi \} \]

Since \((\text{label}^g t) : \{ A \mid \varphi \}\), it follows from Ex. E.31 that we are done if we show

\[ \text{unfold} c (k \circ (g \oplus \ell)^\# \circ (g \oplus r)^\#) : \rightarrow \{ \text{CoList}^g A \mid \square [hd] \varphi \} \]

assuming

\[ k : \rightarrow \{ \text{Rou}^g (\text{CoList}^g A) \mid [\text{Rou}] [hd] \varphi \} \rightarrow \rightarrow \{ \text{CoList}^g A \mid \square [hd] \varphi \} \]

But by Ex. E.53 we are done since

\[ k \circ (g \oplus \ell)^\# \circ (g \oplus r)^\# : \rightarrow \text{Rou}^g (\text{CoList} A) \rightarrow \rightarrow \{ \text{CoList}^g A \mid \square [hd] \varphi \} \]

\[ \square \]

F Proofs of §\ref{7}

Note F.1. In §\ref{F.1}–\ref{F.3} we assume formulae to have no free iteration variables. Free iteration variables in types are then always instantiated in the Adequacy Theorem F.16 (Thm. D.29, Thm. 7.7).

F.1 Correctness of the External and Internal Semantics

Proof of Lem. D.13. (1) (Lem. 7.2)

**Lemma F.2.** If \( \vdash_c \varphi \) in full modal theory of Def. 6.2, then \( \{ [\varphi] \} = \Gamma[A] \).

**Lemma D.19** gives almost all the axioms and rules of Table 2 and Fig. 6 but for the \([\text{ev}(\cdot)]\) modality that we treat separately. We first treat the axioms of Table 2

**Lemma F.3.** If \( \varphi : A \) is an axiom of Table 2, then \( \{ [\varphi] \}^A = [\varphi] \).

**Proof.** Most of the axioms follow from Lem. D.19. Following Def. 6.4 we include the axioms marked (C) in Table 2. The cases of [box] are trivial and omitted.

**Case of (C).** Since in each case, the map \( [[\Delta]] \) preserves \&.

The case of [ev(\cdot)] is treated directly:

\[ \vdash_{B \rightarrow A} ([\text{ev}(\psi)] \psi \land [\text{ev}(\varphi)] \varphi) \implies [\text{ev}(\psi)] (\psi \land \varphi) \]

Let \( x \in \Gamma[B \rightarrow A] \) and assume that \( x \in ([[\text{ev}(\psi)] \psi] \cap [[[\text{ev}(\psi)] \varphi] \varphi]) \). Let now \( y \in \Gamma[B] \) such that \( y \in [\psi] \). We then have \( \text{ev} \circ (x, y) \in [\psi] \cap [\varphi] \).
Case of (N). Since \([[\pi_i]]], [[\text{next}}]] and [[\text{fold}}]] are maps of Heyting algebras. The case of [[\text{ev}}(-)]] is treated directly:

\[ \vdash B \rightarrow A \, [[\text{ev}}(\phi)]] \top \]

Let \( x \in \Gamma[B \rightarrow A] \). Given \( y \in \Gamma[B] \) such that \( y \in [[\phi}}] \), we have \( \text{ev} \circ (x, y) \in \Gamma[A] = \{ \top \} \).

Case of (P). Since \([[\pi_i]]], [[\text{next}}]] and [[\text{fold}}]] are maps of Heyting algebras. As for [[in}}_i], this follows from Lem. D.19.

Case of (C\text{\lor}}). By Lem. D.19.

Case of (C\text{\Rightarrow}}). Since \([[\pi_i]]], [[\text{next}}]] and [[\text{fold}}]] are maps of Heyting algebras. \( \square \)

In order to handle fixpoints, we have the usual monotonicity lemma w.r.t. set inclusion.

**Lemma F.4.** Consider, for a formula \( \alpha_1 : A_1, \ldots, \alpha_k : A_k \vdash \varphi \), the map

\[ [[\varphi}}] : \mathcal{P}(\Gamma[A_1]) \times \cdots \times \mathcal{P}(\Gamma[A_k]) \rightarrow \mathcal{P}(\Gamma[A]), v \mapsto [[\varphi}}]_v \]

For \( i \in \{1, \ldots, k\} \), if \( \alpha_i \text{ Pos } \varphi \) (resp. \( \alpha_i \text{ Neg } \varphi \)), then w.r.t. set inclusion, \([[\varphi}}]] is monotone (resp. anti-monotone) in its \( i \)th argument.

We can now turn to the proof of Lemma F.2.

**Proof (Proof of Lemma F.2).** By induction on \( \vdash A \varphi \). The rules of intuitionistic propositional logic (Fig. 16) as well as of (CL) are trivial and omitted.

Case of

\[ (\text{RM}) \quad \vdash \psi \Rightarrow \varphi \quad \vdash [\Delta] \psi \Rightarrow [\Delta] \varphi \]

By Lem. D.19, this holds for \([\pi_i]], [[\text{next}}]] and [[\text{fold}}]] since \([[\pi_i]]], [[\text{next}}]] and [[\text{fold}}]] are maps of Heyting algebras. As for [[in}}_i], this follows from the fact that \([[\text{in}}_i]] \) preserves implications as it preserves \( \lor \).

The case of [[\text{ev}}(-)]] is treated directly:

\[ \vdash B \rightarrow A \, [[\text{ev}}(\phi)]] \varphi \]

Let \( x \in \Gamma[B \rightarrow A] \). Given \( y \in \Gamma[B] \) such that \( y \in [[\phi}}] \), we have \( \text{ev} \circ (x, y) \in [[\psi}}] \), so that \( \text{ev} \circ (x, y) \in [[\varphi}}] \) since \( [[\psi}}] \subseteq [[\varphi}}] \).

Case of

\[ \vdash A \varphi \quad \vdash [\Box] \varphi \]

Trivial.
Case of

\[ \vdash \Gamma \vdash \phi \quad \vdash \varphi : A \]

Let \( x \in \Gamma[B \to A] \) and assume that \( x \in \{[\text{ev}(\phi)] \varphi \} \). Let furthermore \( y \in \Gamma[B] \) such that \( y \in \{[\psi] \}. \) We have to show \( \text{ev} \circ (x, y) \in \{[\varphi] \}. \) By induction hypothesis we have \( y \in \{[\psi \Rightarrow \phi] \}, \) so that \( y \in \{[\phi] \}. \) But this implies \( \text{ev} \circ (x, y) \in \{[\varphi] \} \) since \( x \in \{[\text{ev}(\phi)] \varphi \} \).

Case of

\[ \vdash \Gamma \vdash \{[\text{ev}(\psi_0)] \varphi \land [\text{ev}(\psi_1)] \varphi \} \Rightarrow [\text{ev}(\psi_0 \lor \psi_1)] \varphi \]

Let \( x \in \Gamma[B \to A] \) and assume that \( x \in \{([\text{ev}(\psi_0)] \varphi \land [\text{ev}(\psi_1)] \varphi) \}. \) Let furthermore \( y \in \Gamma[B] \) such that \( y \in \{[\psi_0 \lor \psi_1] \}. \) We have to show \( \text{ev} \circ (x, y) \in \{[\varphi] \}. \) But if \( y \in \{[\psi_0] \} \) then we are done since \( x \in \{([\text{ev}(\psi_0)] \varphi \}, \) and similarly if \( y \in \{[\psi_1] \}. \)

Case of

\[ \vdash \Gamma \vdash \{[\text{in}_0] \top \lor [\text{in}_1] \top \} \land \lnot([\text{in}_0] \top \land [\text{in}_1] \top) \]

Consider \( x \in \Gamma[A_0 + A_1] \approx \Gamma[A_0] + \Gamma[A_1] \) (via Lem. D.2). Hence \( x = \text{in}_1(y) \) for some \( y \in \Gamma[A_1] \) and we have \( x \in \{[\text{in}_1] \top \}. \) Moreover, since the injections \( \text{in}_0 \) and \( \text{in}_1 \) have disjoint images, we have \( \{[\text{in}_0] \top \land [\text{in}_1] \top \} = \emptyset \) so \( x \in \{\lnot([\text{in}_0] \top \land [\text{in}_1] \top) \}. \)

Case of

\[ \vdash \Gamma \vdash [\text{in}_1] \top \Rightarrow ([\lnot\text{in}_1] \varphi \Leftrightarrow [\text{in}_1] \lnot \varphi) \]

Let \( x \in \Gamma[A_0 + A_1] \approx \Gamma[A_0] + \Gamma[A_1] \), and assume \( x \in \{[\text{in}_1] \top \}, \) so that \( x = \text{in}_1(y) \) for some (unique) \( y \in \Gamma[A_1] \). We show

\( x \in \{[\lnot [\text{in}_1] \varphi \Rightarrow [\text{in}_1] \lnot \varphi] \} \) and \( x \in \{[\text{in}_1] \lnot \varphi \Rightarrow [\text{in}_1] \varphi] \)

For the former, assume \( x \notin \{[\text{in}_1] \varphi \}. \) Since \( y \) is unique such that \( x = \text{in}_1(y) \), we have \( y \notin \{[\varphi] \}. \) But this implies \( y \notin \{[\lnot \varphi] \} \) and we are done.

For the latter, assume \( x \notin \{[\text{in}_1] \lnot \varphi \}. \) Assume toward a contradiction that \( x \in \{[\text{in}_1] \varphi \}. \) Since \( y \) is unique such that \( x = \text{in}_1(y) \), we have both \( y \notin \{[\varphi] \} \) and \( y \notin \{[\varphi] \}, \) a contradiction.

Cases of

\[ \vdash A \mu^0 \alpha \varphi \Leftrightarrow \top \]
\[ \vdash A \mu^{i+1} \alpha \varphi \Leftrightarrow \varphi[\mu^i \alpha \varphi / \alpha] \]
\[ \vdash A \mu^0 \alpha \varphi \Leftrightarrow \bot \]
\[ \vdash A \mu^{i+1} \alpha \varphi \Leftrightarrow [\mu^i \alpha \varphi / \alpha] \]

By definition of \([[\theta^i \alpha \varphi] \].

Cases of

\[ [t] \geq [u] \]
\[ [t] \leq [u] \]

These cases follows from Lem. F.4 (in \( \theta^i \alpha \varphi \) we assume that \( \alpha \) is positive in \( \varphi \)) and the definition of \([[\theta^i \alpha \varphi] \].
Cases of

\[
\begin{align*}
\vdash^A \nu \alpha \varphi & \Rightarrow \varphi [\nu \alpha \varphi / \alpha] & \vdash^A \psi & \Rightarrow \varphi [\psi / \alpha] & \vdash^A \varphi [\mu \alpha \varphi / \alpha] & \Rightarrow \mu \alpha \varphi & \vdash^A \varphi [\psi / \alpha] & \Rightarrow \psi.
\end{align*}
\]

By Lem. [F.4] and the Knaster-Tarski Theorem.

Cases of

\[
\begin{align*}
\vdash^A \mu^\ast \alpha \varphi (\alpha) & \Rightarrow \mu \alpha \varphi (\alpha) & \vdash^A \nu \alpha \varphi (\alpha) & \Rightarrow \nu^\ast \alpha \varphi (\alpha)
\end{align*}
\]

We show by induction on \( m \in \mathbb{N} \) that

\[
\{ [\mu^m \alpha \varphi (\alpha)] \} \subseteq \{ [\mu \alpha \varphi (\alpha)] \} \quad \text{and} \quad \{ [\nu \alpha \varphi (\alpha)] \} \subseteq \{ [\nu^m \alpha \varphi (\alpha)] \}
\]

The base case \( m = 0 \) is trivial since

\[
\{ [\mu^0 \alpha \varphi (\alpha)] \} = \{ [\bot] \} \quad \text{and} \quad \{ [\nu^0 \alpha \varphi (\alpha)] \} = \{ [\top] \}
\]

For the induction step we have

\[
\{ [\mu^{m+1} \alpha \varphi (\alpha)] \} = \{ [\varphi (\mu^m \alpha \varphi (\alpha))] \} \quad \text{and} \quad \{ [\nu^{m+1} \alpha \varphi (\alpha)] \} = \{ [\varphi (\nu^m \alpha \varphi (\alpha))] \}
\]

So the induction hypothesis together with Lem. [F.4] gives

\[
\{ [\mu^{m+1} \alpha \varphi (\alpha)] \} \subseteq \{ [\varphi (\mu \alpha \varphi (\alpha))] \} \quad \text{and} \quad \{ [\varphi (\nu \alpha \varphi (\alpha))] \} \subseteq \{ [\nu^{m+1} \alpha \varphi (\alpha)] \}
\]

and we are done since by the Knaster-Tarski Theorem, we have

\[
\{ [\varphi (\mu \alpha \varphi (\alpha))] \} = \{ [\mu \alpha \varphi (\alpha)] \} \quad \text{and} \quad \{ [\varphi (\nu \alpha \varphi (\alpha))] \} = \{ [\nu \alpha \varphi (\alpha)] \}
\]

\( \square \)

Proof of Lem. [D.13] (2) (Lem. 7.2)

Lemma F.5. If \( \vdash^A \varphi \) in full modal theory of Def. 6.2, then \( [\varphi] = [A] \).

Corollary D.17 gives almost everything we need for the semantic correctness of the modal theory. We begin with the axioms of Table 2.

Lemma F.6. If \( \varphi : A \) is an axiom of Table 2 then \( [\varphi]^A = [A] \).

Proof. Most of the axioms follow from Cor. D.17.

Case of (C). Since in each case, the map \( [\llbracket \triangle \rrbracket] \) preserves \( \land \).

Case of (N). Since in each case, the map \( [\llbracket \triangle \rrbracket] \) preserves \( \top \) (recall that axiom is not assumed for \( [\text{in}_i] \)).

Case of (P). The result for \([\pi_i] \), \([\text{fold}] \) and \([\text{box}] \) follows from the fact that \([\llbracket \pi_i] \), \([\llbracket \text{fold}] \) and \([\llbracket \text{box}] \) are maps of Heyting algebras.

As for \([\text{in}_i] \), it follows from the fact that \([\llbracket \text{in}_i] \) preserves \( \bot \) (Cor. D.17).

Case of (Cv). By Cor. D.17.
Case of \((\text{C}_{\text{op}})\). Since \([[\pi_i]], [[\text{fold}}]]\) and \([[\text{box}]]\) are maps of Heyting algebras.

In order to handle fixpoints, we have the usual monotonicity property w.r.t. subobject posets.

**Lemma F.7.** Consider, for a formula \(\alpha_1 : A_1, \ldots, \alpha_k : A_k \vdash \varphi\), the map

\[
[[\varphi]] : \text{Sub}([[A_1]]) \times \cdots \times \text{Sub}([[A_k]]) \rightarrow \text{Sub}([A]), \nu \mapsto [[\varphi]]_\nu.
\]

For \(i \in \{1, \ldots, k\}\), if \(\alpha_i\) Pos \(\varphi\) (resp. \(\alpha_i\) Neg \(\varphi\)), then w.r.t. subobjects posets, \([[\varphi]]\) is monotone (resp. anti-monotone) in its \(i\)th argument.

We can now turn to the proof of Lemma F.5.

**Proof (Proof of Lemma F.5).** By induction on \(\vdash^A \varphi\). The rules of Fig. 16 follow from the fact that in a topos, the subobjects of a given object form a Heyting algebra.

**Case of**

\[
\frac{\vdash \psi \Rightarrow \varphi}{\vdash [\Delta] \psi \Rightarrow [\Delta] \varphi} \quad \text{(RM)}
\]

The result holds for \([[\pi_i]], [[\text{fold}}]]\) and \([[\text{box}]]\) since \([[\pi_i]]\), \([[\text{fold}}]]\) and \([[\text{box}]]\) are maps of Heyting algebras.

As for \([[\text{in}]], [[\text{next}}]]\) and \([[\text{ev}(-)]]\), this follows from the fact that the maps \([[\text{in}]]\), \([[\text{next}}]]\) and \([[\text{ev}(-)]]\) preserve implications since they preserve \&.

**Case of**

\[
\frac{\vdash^A \varphi}{\vdash_{\text{in}} [[\text{box}]] \varphi} \quad \text{By Cor. D.17}
\]

**Case of**

\[
\frac{\vdash^B \psi \Rightarrow \phi \quad \vdash \varphi : A}{\vdash^{B \rightarrow A} [[\text{ev}([\phi])]] \varphi \Rightarrow [[\text{ev}([\psi])]] \varphi}
\]

This case can be seen as following (via Lem. D.15) from the definition of \([[\text{ev}(-)]]\). A direct argument is nevertheless possible. Let \(t \in [B \rightarrow A](n)\).

Let \(k \leq n\) such that \(t \upharpoonright k \Vdash_k [[\text{ev}([\phi])]] \varphi\). Let furthermore \(\ell \leq k\) and \(u \in [B](\ell)\) such that \(u \Vdash^B \psi\). We have to show \(\text{ev} \circ (t^\upharpoonright \ell, u) \Vdash^A \varphi\). By induction hypothesis we have \(u \Vdash^B \psi \Rightarrow \phi\), so that \(u \Vdash^B \phi\). But this implies \(\text{ev} \circ (t^\upharpoonright \ell, u) \Vdash^A \varphi\) since \(t^\upharpoonright k \Vdash_k [[\text{ev}([\phi])]] \varphi\).

**Case of**

\[
\frac{\vdash^{B \rightarrow A} ( [[\text{ev}([\psi_0])]] \varphi \land [[\text{ev}([\psi_1])]] \varphi \Rightarrow [[\text{ev}([\psi_0] \lor \psi_1)]]] \varphi}{\vdash \text{ev} \circ (t^\upharpoonright \ell, u) \Vdash^A \varphi}. \text{ If } u \Vdash^B \psi_0, \text{ then we are done since } t \Vdash_k [[\text{ev}([\psi_0])]] \varphi, \text{ and similarly if } u \Vdash^B \psi_1.
\]
Case of

\[ \vdash \top \]  
\[ \vdash \top \land \lnot \lnot \top \land \lnot \top \]

Write \( A = A_0 + A_1 \) and consider \( t \in [A_0 + A_1](n) \). Hence \( t = \text{in}_u(u) \) for some \( u \in [A_i](n) \) and we have \( t \Vdash [\text{in}_u]T \). Moreover, since the injections \( \text{in}_0 \) and \( \text{in}_1 \) have disjoint images, we have \( [[\text{in}_0]T \land [\text{in}_1]T](k) = \emptyset \) for all \( k > 0 \) so \( t \Vdash [\text{in}_0]T \land [\text{in}_1]T \).

Case of

\[ \vdash [A_0 + A_1]; [\text{in}_0]T \Rightarrow (\lnot [\text{in}_0]T \land [\text{in}_1]T) \]

Write \( A = A_0 + A_1 \). Let \( t \in [A_0 + A_1](n) \), and let \( k \leq n \) such that \( t \uparrow k \Vdash [\text{in}_0]T \), so that we have \( t \vdash k = \text{in}_u(u) \) for some (unique) \( u \in [A_i](k) \). We show

\[ t \Vdash [A_0 + A_1]; [\text{in}_0]T \Rightarrow [\text{in}_1]T \land [\text{in}_0]T \land [\text{in}_1]T \]

For the former, let \( \ell \leq k \) such that \( t \uparrow \ell = (t \uparrow k) \uparrow \ell \Vdash \lnot [\text{in}_0]T \), that is such that \( t \uparrow \ell \Vdash [\text{in}_0]T \) for all \( m \leq \ell \). We show \( t \uparrow \ell \Vdash [\text{in}_0]T \). Hence we are done if \( u \uparrow m \Vdash [\text{in}_0]T \) for all \( m \leq \ell \). But if \( u \uparrow m \Vdash [\text{in}_0]T \), then we would have \( t \uparrow m = \text{in}_u(u \uparrow m) \Vdash [\text{in}_0]T \), a contradiction.

For the latter, let \( \ell \leq k \) such that \( t \uparrow \ell = (t \uparrow k) \uparrow \ell \Vdash \lnot [\text{in}_1]T \), that is \( t \uparrow \ell \Vdash \lnot [\text{in}_1]T \) for all \( m \leq \ell \). So assume \( t \uparrow \ell \Vdash [\text{in}_1]T \) for some \( m \leq \ell \). Hence, there is \( u' \in [A_i](\ell) \) such that \( t \uparrow \ell = \text{in}_u(u') \) and \( u' \Vdash [\text{in}_1]T \). But we have \( u' = u \uparrow \ell \). On the other hand, since \( t \uparrow \ell \Vdash [\text{in}_1]T \), there is some \( u'' \in [A_i](\ell) \) such that \( t \uparrow \ell = \text{in}_u(u'') \) and \( u'' \Vdash [\text{in}_1]T \) for all \( m \leq \ell \). But we also have \( u'' \uparrow \ell \Vdash [\text{in}_1]T \), thus contradicting \( u \uparrow \ell \Vdash [\text{in}_1]T \).

Cases of

\[ [u] \geq [u] \]
\[ [u] \leq [u] \]

These cases follows from Lem. F.7 (in \( \theta^2 \alpha \varphi \) we assume that \( \alpha \) is positive in \( \varphi \) and the definition of \( [\theta^2 \alpha \varphi] \).

Cases of

\[ [u] \geq [u] \]
\[ [u] \leq [u] \]

By Lem. F.7 and the Knaster-Tarski Theorem, since subobject lattices of \( S \) are complete (Prop. I.8.5)).

Cases of

\[ [u] \geq [u] \]
\[ [u] \leq [u] \]

Similar to the same case in the proof of Lem. F.2
F.2 The Safe Fragment

Lemma F.8 (Lem. D.21). The greatest fixpoint of a Scott cocontinuous function \( f : L \to L \) is given by

\[
\nu(f) := \bigwedge_{n \in \mathbb{N}} f^n(\top)
\]

Proof. That \( \nu(f) \) is a fixpoint of \( f \) follows from the continuity of \( f \) and the fact that the set \( \{ f^n(\top) \mid n \in \mathbb{N} \} \) is codirected, which in turn follows from the fact that \( f \) is monotone. In order to show that \( \nu(f) \) is the greatest fixpoint of \( f \), recall that the greatest fixpoint of \( f \) is in any case given by

\[
b := \bigvee \{ a \in L \mid a \leq f(a) \}
\]

We trivially have \( \nu(f) \leq b \) as \( \nu(f) \) is a fixpoint of \( f \). For the reverse inequality, for all \( a \) such that \( a \leq f(a) \), it follows by induction on \( n \in \mathbb{N} \) and from the monotony of \( f \) that we have \( a \leq f^n(\top) \) for all \( n \in \mathbb{N} \). Hence \( a \leq \nu(f) \) for all \( a \) such that \( a \leq f(a) \), which in turn gives \( b \leq \nu(f) \). \( \square \)

Lemma F.9 (Lem. D.22). Consider a safe formula \( \alpha_1 : P^+_1, \ldots, \alpha_k : P^+_k \Vdash \varphi : P^+ \). The following two functions are Scott-cocontinuous:

\[
\begin{align*}
\{\varphi\} : \text{Sub}(\Gamma P^+_1) \times \cdots \times \text{Sub}(\Gamma P^+_k) & \to \text{Sub}(\Gamma P^+), v \mapsto \{\varphi\}_v \\
\{\varphi\} : \mathcal{P}(\Gamma P^+_1) \times \cdots \times \mathcal{P}(\Gamma P^+_k) & \to \mathcal{P}(\Gamma P^+), v \mapsto \{\varphi\}_v
\end{align*}
\]

Proof. In both cases, monotony w.r.t. lattice order follows by an easy induction from the positivity of safe formulae. We now turn to preservation of codirected meets. We first consider the case of \( \{\varphi\}_v \). We reason by induction on \( \varphi \).

Cases of \( \alpha, \top, \bot \).

Trivial.

Case of \( \varphi \land \psi \).

Let \( D_1 \subseteq \mathcal{P}(\Gamma P^+_1), \ldots, D_k \subseteq \mathcal{P}(\Gamma P^+_k) \) be codirected. By induction hypothesis we obtain

\[
\{\varphi \land \psi\} \bigcap (D_1, \ldots, D_k) = \bigcap \{\varphi\}_v (D_1, \ldots, D_k) \cap \bigcap \{\psi\}_v (D_1, \ldots, D_k)
\]

and the result is trivial.

Case of \( \varphi \lor \psi \).

This is the interesting case. Let \( D_1 \subseteq \mathcal{P}(\Gamma P^+_1), \ldots, D_k \subseteq \mathcal{P}(\Gamma P^+_k) \) be codirected. By induction hypothesis we obtain

\[
\{\varphi \land \psi\} \bigcap (D_1, \ldots, D_k) = \bigcap \{\varphi\}_v (D_1, \ldots, D_k) \cup \bigcap \{\psi\}_v (D_1, \ldots, D_k)
\]

We then trivially get

\[
\bigcap \{\varphi\}_v (D_1, \ldots, D_k) \cup \bigcap \{\psi\}_v (D_1, \ldots, D_k) \subseteq \bigcap \{\varphi \lor \psi\}_v (D_1, \ldots, D_k)
\]
It remains to show the converse direction

$$\bigcap \{\| \varphi \lor \psi \| \} (D_1, \ldots, D_k) \subseteq \bigcap \{\| \varphi \| \} (D_1, \ldots, D_k) \cup \bigcap \{\| \psi \| \} (D_1, \ldots, D_k)$$

So let $$x \in \Gamma^{P^+}$$ such that $$x \in \{\| \varphi \lor \psi \| \} (S_1, \ldots, S_k)$$ for every $$S_1 \subseteq D_1, \ldots, S_k \subseteq D_k$$. Assume toward a contradiction that there are $$S_1 \subseteq D_1, \ldots, S_k \subseteq D_k$$ such that $$x \notin \{\| \varphi \| \} (S_1, \ldots, S_k)$$ and that there are $$S'_1 \subseteq D_1, \ldots, S'_k \subseteq D_k$$ such that $$x \notin \{\| \psi \| \} (S'_1, \ldots, S'_k)$$. Since the $$D_i$$'s are codirected for inclusion, there are $$S''_1 \subseteq D_1, \ldots, S''_k \subseteq D_k$$ such that $$S''_i \subseteq S_i \cap S'_i$$ for $$i = 1, \ldots, k$$. By monotonicity w.r.t. inclusion, we have $$x \notin \{\| \varphi \| \} (S''_1, \ldots, S''_k)$$ and $$x \notin \{\| \psi \| \} (S''_1, \ldots, S''_k)$$. But this implies $$x \notin \{\| \varphi \lor \psi \| \} (S''_1, \ldots, S''_k)$$, a contradiction.

**Case of** $$[\pi]_i \varphi$$.

Let $$D_1 \subseteq \mathcal{P}(\Gamma^{P^+})$$, $$\ldots, D_k \subseteq \mathcal{P}(\Gamma^{P^+})$$ be codirected. Let $$x \in \Gamma^{P^+}$$ and write $$P^+ = Q_0^+ \times Q_1^+$$. Then we are done since by induction hypothesis

$$x \in \{[\pi]_i \varphi\} (\bigcap D_1, \ldots, \bigcap D_k)$$

if $$\pi_i \circ x \in \{\| \varphi \| \} (\bigcap D_1, \ldots, \bigcap D_k)$$

if $$\forall S_1 \subseteq D_1, \ldots, S_k \subseteq D_k, \; \pi_i \circ x \in \{\| \varphi \| \} (D_1, \ldots, D_k)$$

if $$\forall S_1 \subseteq D_1, \ldots, S_k \subseteq D_k, \; x \in \{[\pi]_i \varphi\} (D_1, \ldots, D_k)$$

if $$x \in \bigcap \{[\pi]_i \varphi\} (D_1, \ldots, D_k)$$

**Case of** $$[\text{in}]_i \varphi$$.

Let $$D_1 \subseteq \mathcal{P}(\Gamma^{P^+})$$, $$\ldots, D_k \subseteq \mathcal{P}(\Gamma^{P^+})$$ be codirected. Let $$x \in \Gamma^{P^+}$$ and write $$P^+ = Q_0^+ \times Q_1^+$$. By Lem. D.2, we have $$x = \text{in}_j \circ y$$ for some unique $$j \in \{0, 1\}$$ and $$y \in \Gamma^Q$$. Then we are done since by induction hypothesis we have $$x \in \{[\text{in}]_i \varphi\} (\bigcap D_1, \ldots, \bigcap D_k)$$

if $$j = i$$ and $$y \in \{\| \varphi \| \} (\bigcap D_1, \ldots, \bigcap D_k)$$

if $$j = i$$ and $$y \in \bigcap \{\| \varphi \| \} (D_1, \ldots, D_k)$$

if $$j = i$$ and $$\forall S_1 \subseteq D_1, \ldots, S_k \subseteq D_k, \; y \in \{\| \varphi \| \} (D_1, \ldots, D_k)$$

if $$\forall S_1 \subseteq D_1, \ldots, S_k \subseteq D_k, \; x \in \{[\text{in}]_i \varphi\} (D_1, \ldots, D_k)$$

if $$x \in \bigcap \{[\text{in}]_i \varphi\} (D_1, \ldots, D_k)$$

**Case of** $$[\text{next}] \varphi$$.

Let $$D_1 \subseteq \mathcal{P}(\Gamma^{P^+})$$, $$\ldots, D_k \subseteq \mathcal{P}(\Gamma^{P^+})$$ be codirected. Let $$x \in \Gamma^{P^+}$$ and write $$P^+ = \nabla Q^+$$ and $$Q^+ = Q_0^+ \times Q_1^+$$. By Lem. D.2, we have $$x = \text{next} \circ y$$ for some unique $$y \in \Gamma^Q$$. Then we are done since by induction hypothesis we have

$$x \in \{[\text{next}] \varphi\} (\bigcap D_1, \ldots, \bigcap D_k)$$

if $$y \in \{\| \varphi \| \} (\bigcap D_1, \ldots, \bigcap D_k)$$

if $$y \in \bigcap \{\| \varphi \| \} (D_1, \ldots, D_k)$$

if $$\forall S_1 \subseteq D_1, \ldots, S_k \subseteq D_k, \; y \in \{\| \varphi \| \} (D_1, \ldots, D_k)$$

if $$\forall S_1 \subseteq D_1, \ldots, S_k \subseteq D_k, \; x \in \{[\text{next}] \varphi\} (D_1, \ldots, D_k)$$

if $$x \in \bigcap \{[\text{next}] \varphi\} (D_1, \ldots, D_k)$$

**Case of** $$[\text{fold}] \varphi$$.

This case is dealt with similarly as that of $$[\pi]_i$$. 113
**Case of** \([\text{box}]\varphi\).

Trivial since \(\varphi\) is required to be closed.

**Case of** \([\text{ev}](\psi)\] \(\varphi\).

Note that \(\psi\) is assumed to be closed since \([\text{ev}](\psi)\] \(\varphi\) is safe. Let \(D_1 \subseteq \mathcal{P}(\Gamma[P^+_1]), \ldots, D_k \subseteq \mathcal{P}(\Gamma[P^+_k])\) be codirected. Let \(x \in \Gamma[P^+]\) and write \(P^+ = R^+ \rightarrow Q^+\). Then we are done since by induction hypothesis we have

\[
x \in [[\text{ev}(\psi)](\varphi)](\bigcap D_1, \ldots, \bigcap D_k)
\]

if \(\forall y \in [[\psi]], \text{ev} \circ (x, y) \in [[\varphi]](\bigcap D_1, \ldots, \bigcap D_k)\)

if \(\forall y \in [[\psi]], \text{ev} \circ (x, y) \in [[\varphi]](D_1, \ldots, D_k)\)

if \(\forall S_1 \in D_1, \ldots, S_k \in D_k, \forall y \in [[\psi]], \text{ev} \circ (x, y) \in [[\varphi]](S_1, \ldots, S_k)\)

if \(\forall S_1 \in D_1, \ldots, S_k \in D_k, x \in [[\text{ev}(\psi)](\varphi)](S_1, \ldots, S_k)\)

if \(x \in \bigcap [[\text{ev}(\psi)](\varphi)](D_1, \ldots, D_k)\)

**Cases of** \(\theta^\nu \alpha \varphi\).

By induction hypothesis, the function

\[
[[\varphi]] : \mathcal{P}(\Gamma[P^+_1]) \times \cdots \times \mathcal{P}(\Gamma[P^+_k]) \times \mathcal{P}(\Gamma[P^+]) \rightarrow \mathcal{P}(\Gamma[P^+]), \quad v, S \mapsto [[\varphi]]_{v[S/\alpha]}
\]

is Scott-cocontinuous. Hence by Lem. \textbf{F.8} for \(S_1 \in \mathcal{P}(\Gamma[P^+_1]), \ldots, S_k \in \mathcal{P}(\Gamma[P^+_k])\) we have

\[
[[\nu^\alpha \varphi]](S_1, \ldots, S_k) = ([[\varphi]](S_1, \ldots, S_k))^{m}(\top)
\]

where

\[
[[\varphi]](S_1, \ldots, S_k)^{m+1}(\top) = [[\varphi]](S_1, \ldots, S_k, ([[\varphi]](S_1, \ldots, S_k))^{m}(\top))
\]

and where \(([[\varphi]](S_1, \ldots, S_k))^{0}(\top) := \top\) and \(([[\varphi]](S_1, \ldots, S_k))^{0}(\bot) := \bot\).

An easy induction on \(m \in \mathbb{N}\) then shows that each function

\[
([[\varphi]](\text{--}, \ldots, \text{--}))^{m}(\top) : \mathcal{P}(\Gamma[P^+_1]) \times \cdots \times \mathcal{P}(\Gamma[P^+_k]) \rightarrow \mathcal{P}(\Gamma[P^+])
\]

is Scott-cocontinuous.

**Case of** \(\nu \alpha \varphi\).

By induction hypothesis, the function

\[
[[\varphi]] : \mathcal{P}(\Gamma[P^+_1]) \times \cdots \times \mathcal{P}(\Gamma[P^+_k]) \times \mathcal{P}(\Gamma[P^+]) \rightarrow \mathcal{P}(\Gamma[P^+]), \quad v, S \mapsto [[\varphi]]_{v[S/\alpha]}
\]

is Scott-cocontinuous. Hence by Lem. \textbf{F.8} for \(S_1 \in \mathcal{P}(\Gamma[P^+_1]), \ldots, S_k \in \mathcal{P}(\Gamma[P^+_k])\) we have

\[
[[\nu \alpha \varphi]](S_1, \ldots, S_k) = \bigcap_{n \in \mathbb{N}} ([[\varphi]](S_1, \ldots, S_k))^{n}(\top)
\]

where \(([[\varphi]](S_1, \ldots, S_k))^{0}(\top) := \top\) and

\[
([[\varphi]](S_1, \ldots, S_k))^{n+1}(\top) = [[\varphi]](S_1, \ldots, S_k, ([[\varphi]](S_1, \ldots, S_k))^{n}(\top))
\]
An easy induction on $n$ shows that each function

$$(\{\nu\alpha.\varphi\}(-,\ldots, -))^n(\top) : \mathcal{P}(\Gamma[P^+]) \times \cdots \times \mathcal{P}(\Gamma[P^+_k]) \rightarrow \mathcal{P}(\Gamma[P^+])$$

is Scott-cocontinuous.

Consider now codirected $D_1 \subseteq \mathcal{P}(\Gamma[P^+_1]), \ldots, D_k \subseteq \mathcal{P}(\Gamma[P^+_k])$. Then we are done since

$$\{\nu\alpha.\varphi\}(\bigcap D_1, \ldots, \bigcap D_k) = \bigcap_{n \in \mathbb{N}}\{\nu\alpha.\varphi\}(\bigcap D_1, \ldots, \bigcap D_k)^n(\top)$$

We now turn to the case of $\nu\alpha.\varphi$. This case cannot occur since $\nu\alpha.\varphi$ is not safe.

We now turn to the case of $\nu\alpha.\varphi$. This case cannot occur since $\nu\alpha.\varphi$ is not safe.

\textbf{Case of } $\nu\alpha.\varphi$. This case cannot occur since $\nu\alpha.\varphi$ is not safe.

\textbf{Case of } $\nu\alpha.\varphi$. This case cannot occur since $\nu\alpha.\varphi$ is not safe.

\begin{align*}
\nu\alpha.\varphi & : \text{Sub}(\Gamma[P^+_1]) \times \cdots \times \text{Sub}(\Gamma[P^+_k]) \rightarrow \text{Sub}(\Gamma[P^+])
\end{align*}

being Scott-continuous means that for $D_1 \subseteq \text{Sub}(\Gamma[P^+_1]), \ldots, D_k \subseteq \text{Sub}(\Gamma[P^+_k])$ codirected w.r.t. subobject lattice orders, we have

$$\nu\alpha.\varphi(\bigwedge D_1, \ldots, \bigwedge D_k) = \bigwedge \nu\alpha.\varphi(D_1, \ldots, D_k)$$

But since meets in subobject lattices of $\mathcal{S}$ are pointwise, the above is equivalent to have, for all $n > 0$ that

$$\nu\alpha.\varphi(\bigwedge D_1, \ldots, \bigwedge D_k)(n) = \bigcap \nu\alpha.\varphi(D_1, \ldots, D_k)(n)$$

\textbf{Cases of } $\alpha$, $\top$, $\bot$.

\textbf{Trivial.}

\textbf{Case of } $\varphi \land \psi$.

Let $D_1 \subseteq \text{Sub}(\Gamma[P^+_1]), \ldots, D_k \subseteq \text{Sub}(\Gamma[P^+_k])$ be codirected. By induction hypothesis we obtain

$$\varphi \land \psi(\bigwedge D_1, \ldots, \bigwedge D_k) = \bigwedge \varphi(D_1, \ldots, D_k) \land \bigwedge \psi(D_1, \ldots, D_k)$$

and the result is trivial.

\textbf{Case of } $\varphi \lor \psi$.

Let $D_1 \subseteq \text{Sub}(\Gamma[P^+_1]), \ldots, D_k \subseteq \text{Sub}(\Gamma[P^+_k])$ be codirected. By induction hypothesis we obtain

$$\varphi \land \psi(\bigwedge D_1, \ldots, \bigwedge D_k) = \bigwedge \varphi(D_1, \ldots, D_k) \lor \bigwedge \psi(D_1, \ldots, D_k)$$

By monotonicity w.r.t. subobject lattice orders, we trivially get

$$\bigwedge \varphi(D_1, \ldots, D_k) \lor \bigwedge \psi(D_1, \ldots, D_k) \subseteq \bigwedge \varphi \lor \psi(D_1, \ldots, D_k)$$
It remains to show the converse direction
\[
\bigwedge [\varphi \lor \psi](D_1, \ldots, D_k) \subseteq \bigwedge [\varphi](D_1, \ldots, D_k) \lor [\psi](D_1, \ldots, D_k)
\]
Since meets and joins are computed pointwise in subobject lattices, we are done if for each \( n > 0 \) we show
\[
\bigcap [\varphi \lor \psi](D_1, \ldots, D_k)(n) \subseteq \bigcap [\varphi](D_1, \ldots, D_k)(n) \lor \bigcap [\psi](D_1, \ldots, D_k)(n)
\]
We can then conclude as in the case of \([-\cdot]\). Fix \( n > 0 \) and let \( t \in [P^+] \) such that \( t \notin [\varphi \lor \psi](A_1, \ldots, A_k)(n) \) for every \( A_1 \in D_1, \ldots, A_k \in D_k \). Assume toward a contradiction that there are \( A_1 \in D_1, \ldots, A_k \in D_k \) such that \( t \notin [\varphi](A_1, \ldots, A_k)(n) \) and that there are \( A'_1 \in D_1, \ldots, A'_k \in D_k \) such that \( t \notin [\psi](A'_1, \ldots, A'_k)(n) \). Since the \( D_i \)'s are codirected for inclusion, there are \( A''_1 \in D_1, \ldots, A''_k \in D_k \) such that \( A''_i \leq A_i \) for \( i = 1, \ldots, k \). By monotonicity w.r.t. subobject lattice orders, we have \( t \notin [\varphi](A''_1, \ldots, A''_k)(n) \) and \( t \notin [\psi](A''_1, \ldots, A''_k)(n) \). But this implies \( t \notin [\varphi \lor \psi](A''_1, \ldots, A''_k)(n) \), a contradiction.

**Case of \([\pi_i]_{\varphi}\).**
Let \( D_1 \subseteq \text{Sub}([P^+]) \), \( D_k \subseteq \text{Sub}([P^+]) \) be codirected. We show that for all \( n > 0 \) we have
\[
[[\pi_i]_{\varphi}](\bigwedge D_1, \ldots, \bigwedge D_k)(n) = \bigcap [\varphi](D_1, \ldots, D_k)(n)
\]
and this goes similarly as for \([-\cdot]\).

**Case of \([\text{in}_i]_{\varphi}\).**
Let \( D_1 \subseteq \text{Sub}([P^+]) \), \( D_k \subseteq \text{Sub}([P^+]) \) be codirected. We show that for all \( n > 0 \) we have
\[
[[\text{in}_i]_{\varphi}](\bigwedge D_1, \ldots, \bigwedge D_k)(n) = \bigcap [\varphi](D_1, \ldots, D_k)(n)
\]
and this goes similarly as for \([-\cdot]\) since the pointwise maps \( (\text{in}_i)_n : [Q^+](n) \to [Q^+](n) \) and \( [Q^+](n) \) are injective.

**Case of \([\text{next}]_{\varphi}\).**
Let \( D_1 \subseteq \text{Sub}([P^+]) \), \( D_k \subseteq \text{Sub}([P^+]) \) be codirected. Write \( P^+ = \mathbf{Q}^+ \). We show that for all \( n > 0 \) we have
\[
[[\text{next}]_{\varphi}](\bigwedge D_1, \ldots, \bigwedge D_k)(n) = \bigcap [[\text{next}]_{\varphi}](D_1, \ldots, D_k)(n)
\]
The result is trivial if \( n = 1 \). For \( n > 1 \), it reduces to
\[
[[\varphi](\bigwedge D_1, \ldots, \bigwedge D_k)(n - 1) = \bigcap [[\varphi](D_1, \ldots, D_k)(n - 1)
\]
which follows from the induction hypothesis.

**Case of \([\text{fold}]_{\varphi}\).**
This case is handled similarly as that of \([\pi_i]\).

**Case of \([\text{box}]_{\varphi}\).**
Trivial since \( \varphi \) is required to be closed.
Case of $[\text{ev}(\psi)]\varphi$.

Note that $\psi$ is assumed to be closed since $[\text{ev}(\psi)]\varphi$ is safe. Let $D_1 \subseteq \text{Sub}(\{P_1^+\}), \ldots, D_k \subseteq \text{Sub}(\{P_k^+\})$ be codirected. Write $P^+ = R^+ \rightarrow Q^+$. We show that for all $n > 0$ we have

$$[[\text{ev}(\psi)]\varphi](\bigwedge D_1, \ldots, \bigwedge D_k)(n) = \bigcap[[\text{ev}(\psi)]\varphi](D_1, \ldots, D_k)(n)$$

Let $n > 0$ and $t \in [[P^+]](n)$. Then we are done since by induction hypothesis we have:

1. If $t \in [[\text{ev}(\psi)]\varphi](\bigwedge D_1, \ldots, \bigwedge D_k)(n)$
2. If $t \leq n$, $\forall u \in [[\psi](\ell)]$, $\text{ev} \circ (t\uparrow\ell, u) \in [[\varphi](\bigwedge D_1, \ldots, \bigwedge D_k)(\ell)]$
3. If $t \leq n$, $\forall u \in [[\psi](\ell)]$, $\text{ev} \circ (t\uparrow\ell, u) \in \bigcap[[\varphi](D_1, \ldots, D_k)(\ell)]$
4. If $t \leq n$, $\forall u \in [[\psi](\ell)]$, $\text{ev} \circ (t\uparrow\ell, u) \in [[\varphi](S_1, \ldots, S_k)(\ell)]$
5. If $t \leq n$, $\forall u \in [[\psi](\ell)]$, $\text{ev} \circ (t\uparrow\ell, u) \in [[\varphi](S_1, \ldots, S_k)(\ell)]$
6. If $t \leq n$, $\forall u \in [[\psi](\ell)]$, $\text{ev} \circ (t\uparrow\ell, u) \in [[\varphi](S_1, \ldots, S_k)(\ell)]$

Cases of $\theta^\alpha \varphi$ and $\nu \alpha \varphi$.

These cases are handled exactly as for $\{|-\|\}$.

Case of $\nu \alpha \varphi$.

This case cannot occur since $\mu \alpha \varphi$ is not safe. $\square$

Proposition F.10 (Prop. 7.3). Let $\alpha_1 : P_1^+, \ldots, \alpha_k : P_k^+ \vdash \varphi : P^+$ be a safe formula. Given $S_1 \subseteq \text{Sub}(\{P_1^+\}), \ldots, S_k \subseteq \text{Sub}(\{P_k^+\})$, we have

$$[\varphi](\Gamma(S_1), \ldots, \Gamma(S_k)) = \Gamma([\varphi](S_1, \ldots, S_k))$$

Proof. We reason by induction on the derivation of $\alpha_1 : P_1^+, \ldots, \alpha_k : P_k^+ \vdash \varphi : P^+$. In all cases but $\theta^\alpha \varphi$ and $\nu \alpha \varphi$, the parameters are irrelevant and we omit them.

Cases of $\alpha$, $\top$ and $\bot$.

Trivial.

Case of $\varphi \land \psi$.

Let $x \in \Gamma[P^+]$. Then we are done since by induction hypothesis we have

1. If $x \in [[\varphi \land \psi]]$ and $x \in [[\varphi]]$
2. If $\forall n > 0$, $x_n(\bullet) \in [[\varphi]](n)$ and $x_n(\bullet) \in [[\psi]](n)$
3. If $\forall n > 0$, $x_n(\bullet) \in [[\varphi]](n)$ and $x_n(\bullet) \in [[\psi]](n)$
4. If $\forall n > 0$, $x_n(\bullet) \in [[\varphi \land \psi]](n)$

Case of $\varphi \lor \psi$.

Let $x \in \Gamma[P^+]$. Assume first that $x \in [[\varphi \lor \psi]]$. If (say) $x \in [[\varphi]]$, then by induction hypothesis we get $x_n(\bullet) \in [[\varphi]](n)$ for all $n > 0$, which implies $x_n(\bullet) \in [[\varphi \lor \psi]](n)$ for all $n > 0$.

Conversely, assume that $x_n(\bullet) \in [[\varphi \lor \psi]](n)$ for all $n > 0$. Assume toward a contradiction that there are $k, \ell > 0$ with (say) $k \leq \ell$ such that $x_k(\bullet) \notin [[\varphi]](n)$ and $x_\ell(\bullet) \notin [[\psi]](n)$. Since $k \leq \ell$, by Lem. D.16 we have $x_k(\bullet) \notin [[\psi]](n)$, but this contradicts $x_k(\bullet) \in [[\varphi \lor \psi]](n)$. Hence, we have either $x_n(\bullet) \in [[\varphi]](n)$ for all $n > 0$ or $x_n(\bullet) \in [[\psi]](n)$ for all $n > 0$, and the result follows by induction hypothesis.
Case of $\psi \Rightarrow \varphi$.

This case cannot occur since $\psi \Rightarrow \varphi$ is not safe.

Case of $[\pi_i] \varphi$.

Let $x \in \Gamma[P^+]$ and write $P^+ = Q_0^+ \times Q_1^+$. Then we are done since $(\pi_i \circ x)_n(\bullet) = \pi_i(x_n(\bullet))$ so that by induction hypothesis we have

$$x \in \{[\pi_i] \varphi\} \text{ iff } \pi_i \circ x \in \{\varphi\}$$

$$\text{iff } \forall n > 0, \ (\pi_i \circ x)_n(\bullet) \in \{\varphi\}(n)$$

$$\text{iff } \forall n > 0, \ x_n(\bullet) \in \{[\pi_i] \varphi\}(n)$$

Case of $[in_j] \varphi$.

Let $x \in \Gamma[P^+]$ and write $P^+ = Q_0^+ + Q_1^+$. By Lem. D.2 we have $x = in_j \circ y$ for some unique $j \in \{0, 1\}$ and $y \in \Gamma[Q_j^+]$. Then we are done since $x_n(\bullet) = (in_j \circ y)_n(\bullet) = in_j(y_n(\bullet))$ so that by induction hypothesis we have

$$x \in \{[in_j] \varphi\} \text{ iff } j = i \text{ and } y \in \{\varphi\}$$

$$\text{iff } j = i \text{ and } \forall n > 0, \ y_n(\bullet) \in \{\varphi\}(n)$$

$$\text{iff } \forall n > 0, \ x_n(\bullet) \in \{[in_j] \varphi\}(n)$$

Case of $[next] \varphi$.

Let $x \in \Gamma[P^+]$ and write $P^+ = \triangleright Q^+$. By Lem. D.2 we have $x = next \circ y$ for some unique $y \in \Gamma[Q^+]$. Assume first $x \in \{[next] \varphi\}$. Hence we have

$$y \in \{\varphi\}, \text{ which by induction hypothesis implies } y_n(\bullet) \in \{\varphi\}(n) \text{ for all } n > 0.$$ 

Now, we trivially have $x_1(\bullet) \in \{[next] \varphi\}(1)$. Moreover, for $n > 1$, we have $x_n(\bullet) = y_{n-1}(\bullet)$, so that $x_n(\bullet) \in \{[next] \varphi\}(n) = \{\varphi\}(n-1)$. Assume conversely that $x_n(\bullet) \in \{[next] \varphi\}(n)$ for all $n > 0$. This implies $x_n(\bullet) \in \{\varphi\}(n-1)$ for all $n > 1$, which in turn implies $y_{n-1}(\bullet) \in \{\varphi\}(n-1)$ for all $n > 1$. But by induction hypothesis this implies $y \in \{\varphi\}$ so that $x \in \{[next] \varphi\}$.

Case of $[fold] \varphi$.

This case is handled similarly as that of $[\pi_i]$.

Case of $[box] \varphi$.

Recall that $\varphi$ is required to be closed. Also, by definition we have

$$\{[box] \varphi\}^A(n) := \{t \in \{[box] A\}(n) = \Gamma[A] \mid t \in \{\varphi\}^A\}$$

$$\{[box] \varphi\}^A := \{x \in \Gamma[\{box] A] \mid x(\bullet) \in \{\varphi\}^A\}$$

It follows that given $x \in \Gamma[\{box] A]$, we have

$$x \in \{[box] \varphi\}^A \text{ iff } x_1(\bullet) \in \{\varphi\}^A$$

$$\text{iff } \forall n > 0, \ x_n(\bullet) \in \{\varphi\}^A$$

$$\text{iff } \forall n > 0, \ x_n(\bullet) \in \{[box] \varphi\}^A(n)$$

Case of $[ev(\psi)] \varphi$.

This case cannot occur since $P^+$ is assumed to be strictly positive.
Case of $\theta_{\alpha}^\varphi$.
Since $[\textbf{ev}(\psi)]\varphi$ is smooth, the formula $\psi$ is closed and we have $Q^+ = B \rightarrow R^+$ where $B$ is constant. Since $B$ is constant, by Lem. [D,4] there is a set $A$ such that $[B] \simeq \Delta A$, so that $[\Gamma[B]] \simeq A$ by Lem. [D,2]. Moreover, it follows from Lem. [D,24] that $[\psi]$ is also constant, so there is a set $S$ such that $[\psi] \simeq \Delta S$.

Now, by induction hypothesis we have $\Gamma[\psi] = \{|\psi|\}$. Since $\Gamma[\Delta] \simeq \textbf{Id}_{\text{Set}}$ (Lem. [D,2]), it follows that $[\psi] \simeq \Delta[\psi]$. We then have

$$x \in [\textbf{ev}(\psi)]\varphi \text{ iff } \forall y \in \Gamma'[B] (y \in \{|\psi|\} \implies \textbf{ev} \circ \langle x, y \rangle \in \{|\varphi|\})$$

and

$$t \in [\textbf{ev}(\psi)]\varphi(n) \text{ iff } \forall k \leq n, \forall u \in A (u \in \{|\psi|\} \implies (t\uparrow k)(u) \in \{|\varphi|\}(k))$$

Given $x \in \Gamma[B \rightarrow R^+]$ and $y \in \Gamma'[B]$, for all $0 \leq k \leq n$ we have

$$\textbf{ev} \circ \langle x, y \rangle)_n^k = (x_n^k)(y_k^k)$$

Since $[\varphi] \simeq \Delta[\varphi]$ by induction hypothesis, we are done with

$$x \in [\textbf{ev}(\psi)]\varphi \text{ iff } \forall y \in \Gamma'[B] (y \in \{|\psi|\} \implies \textbf{ev} \circ \langle x, y \rangle \in \{|\varphi|\})$$

and

$$\forall n \geq 0, \forall u \in A (u \in \{|\psi|\} \implies \forall k \leq n, (\textbf{ev} \circ \langle x, y \rangle)_n^k \in \{|\varphi|\}(k))$$

Case of $\mu\alpha\varphi$.

Assume $P_1^+, \ldots, P_k^+, \alpha : P^+ \vdash \varphi(\alpha) : P^+$, and let $S_1 \in \text{Sub}(\{P_1^+\}), \ldots, S_k \in \text{Sub}(\{P_k^+\})$. Using the induction hypothesis on $\varphi$, an easy induction on $m \in \mathbb{N}$ shows that

$$\{\varphi^m\} (\Gamma(S_1), \ldots, \Gamma(S_k), \top) = \Gamma\{\varphi^m(S_1, \ldots, S_k, \top)\}$$

Case of $\nu\alpha\varphi$.

Assume $P_1^+, \ldots, P_k^+, \alpha : P^+ \vdash \varphi(\alpha) : P^+$, and let $S_1 \in \text{Sub}(\{P_1^+\}), \ldots, S_k \in \text{Sub}(\{P_k^+\})$. Similarly as above, for all $m \in \mathbb{N}$ we have

$$\{\varphi^m\} (\Gamma(S_1), \ldots, \Gamma(S_k), \top) = \Gamma\{\varphi^m(S_1, \ldots, S_k, \top)\}$$

It then directly follows that for all $x \in \Gamma[P^+]$, we have

$$x \in \bigcap_{m \in \mathbb{N}}\{\varphi^m\} (\Gamma(S_1), \ldots, \Gamma(S_k), \top)$$

if $\forall n \geq 0, x_n^k \in \bigcap_{m \in \mathbb{N}}\{\varphi^m\}(S_1, \ldots, S_k, \top)(n)$

and we conclude by Lem. [F,9] and Lem. [F,8].

Case of $\mu\alpha\varphi$.

This case cannot occur since $\mu\alpha\varphi$ is not safe.
F.3 The Smooth Fragment

Assume for this § that the set of propositional variables is partitioned into two infinite sets \( \{ \alpha, \beta, \ldots \} \) and \( \{ \alpha^\mu, \beta^\mu, \ldots \} \) of respectively gfp (or \( \nu \)) and lfp (or \( \mu \)) propositional variables. Write \( \Sigma^\nu \) (resp. \( \Sigma^\mu \)) if the context \( \Sigma \) only declares gfp (resp. lfp) propositional variables.

Lemma F.11. If \( \varphi \) is alternation-free, then \( \varphi \) can be formed with the rules of Fig. 5 and Fig. 9, but with the rules (\( \nu \)-F) and (\( \mu \)-F) replaced respectively by

\[
\frac{\Sigma^\nu, \alpha^\nu : A \vdash \varphi : A}{\Sigma^\nu \vdash \nu \alpha^\nu \varphi : A} \quad \frac{\Sigma^\mu, \alpha^\mu : A \vdash \varphi : A}{\Sigma^\mu \vdash \mu \alpha^\mu \varphi : A}
\]

where in both cases \( \alpha^\theta \) is guarded in \( \varphi \), and \( \alpha^\theta \) as well as all variables of \( \Sigma^\theta \) are positive in \( \varphi \).

Proof. By induction on \( \varphi \). The only relevant cases are \( \theta \alpha \varphi \). Since the two cases are similar, we only discuss that of \( \Sigma \vdash \nu \alpha \varphi : A \).

Assume toward a contradiction that \( \Sigma \) cannot be made empty. This means that there is some variable \( \beta^\mu \) which does occur in \( \varphi \), and such that \( \beta^\mu \) must occur in the context of a \( \mu \) rule for some subformula of \( \varphi \). But then \( \beta^\mu \) occurs free in \( \nu \alpha \varphi \) under two fixpoints of different kinds, a contradiction. It follows that we can assume \( \Sigma \) empty. Similarly, \( \alpha \) can be assumed to be gfp variable, since otherwise it would occur free under a lfp in \( \nu \alpha \varphi \).

\[\blacksquare\]

Lemma F.12 (Lem. 7.4). Let \( \alpha_1 : P_1^+, \ldots, \alpha_k : P_k^+, \alpha : Q^+ \vdash \varphi : P^+ \) be a smooth formula and let \( v \) be a valuation taking each propositional variable \( \alpha_i \) for \( i = 1, \ldots, k \) to a set \( v(\alpha_i) \in P(\Gamma[P_i^+]) \). Consider the function

\[
\{ | \varphi | \} : P(\Gamma[Q^+]) \rightarrow P(\Gamma[P^+]), S \mapsto \{|\varphi|\}_v[S/\alpha]
\]

Then,

- if \( \alpha \) is positive in \( \varphi \) (i.e. \( \alpha \Pos \varphi \)):
  - if \( \alpha \) is a gfp variable, then \( \{|\varphi|\} \) is Scott-cocontinuous,
  - if \( \alpha \) is a lfp variable, then \( \{|\varphi|\} \) is Scott-continuous,
- if \( \alpha \) is negative in \( \varphi \) (i.e. \( \alpha \Neg \varphi \)), then \( \{|\varphi|\} \) is antimonotone and
  - takes meets of codirected sets to joins of directed sets if \( \alpha \) is a gfp variable,
  - takes joins of directed sets to meets of codirected sets if \( \alpha \) is a lfp variable.

Proof. The proof is by induction on formation of formulae \( \alpha_1 : P_1^+, \ldots, \alpha_k : P_k^+, \alpha : Q^+ \vdash \varphi : P^+ \). Monotonicity and antimonotonicity follow from Lem. F.4. Note that since formulae of the form \( [\boxdot] \varphi \) are necessarily closed, nothing has to be proved for these. Some cases are already handled by Lem. D.22 (Lem. F.4), and we do not repeat them. We omit the valuation \( v \) when possible.
Cases of $\alpha, \top, \bot$.

Trivial.

Case of $\varphi \land \psi$ (monotone).

Preservation of codirected meets is trivial (see Lem. D.22 (Lem. F.9)). As for the preservation of directed joins, let $D \subseteq \mathcal{P}(\Gamma[Q^+])$ be directed. Then by induction hypothesis we have

$$\{\varphi \land \psi\} \bigcup D = \bigcup \{\varphi\} (D) \cap \bigcup \{\psi\} (D) \supseteq \bigcup \{\varphi \land \psi\} (D)$$

For the converse inclusion, consider some $x$ both in $\bigcup \{\varphi\} (D)$ and $\bigcup \{\psi\} (D)$. Hence there are $S, S' \in D$ such that $x \in \{\varphi\} (S)$ and $x \in \{\psi\} (S')$. Now since $D$ is directed and by monotonicity, there is some $S'' \in D$ such that $x \in \{\varphi\} (S'') \cap \{\psi\} (S'')$.

Case of $\varphi \land \psi$ (antimonotone).

That $\{\varphi \land \psi\}$ turns directed joins into codirected meets is trivial (as directed meets commute over binary meets) and omitted. Let us show that $\{\varphi \land \psi\}$ turns codirected meets into directed joins. So let $D \subseteq \mathcal{P}(\Gamma[Q^+])$ be codirected. Then by induction hypothesis we have

$$\{\varphi \land \psi\} \bigcap D = \bigcup \{\varphi\} (D) \cap \bigcup \{\psi\} (D) \supseteq \bigcup \{\varphi \land \psi\} (D)$$

We then conclude as for preservation of directed joins in the monotone case. Given $x$ both in $\bigcup \{\varphi\} (D)$ and $\bigcup \{\psi\} (D)$, there are $S, S' \in D$ such that $x \in \{\varphi\} (S)$ and $x \in \{\psi\} (S')$. Now since $D$ is codirected there is some $S'' \in D$ such that $S'' \subseteq S \cap S'$, and by antimonotonicity we have $x \in \{\varphi\} (S'') \cap \{\psi\} (S'')$.

Case of $\varphi \lor \psi$ (monotone).

Preservation of codirected meets is handled in Lem. D.22 (Lem. F.9) while preservation of directed join is trivial.

Case of $\varphi \lor \psi$ (antimonotone).

That $\{\varphi \lor \psi\}$ turns codirected meets into directed joins is trivial (as codirected meets commute over binary meets) and omitted. Let us show that $\{\varphi \lor \psi\}$ turns directed joins into codirected meets. So let $D \subseteq \mathcal{P}(\Gamma[Q^+])$ be directed. By induction hypothesis we have

$$\{\varphi \lor \psi\} \bigcap D = \bigcap \{\varphi\} (D) \cup \bigcap \{\psi\} (D) \subseteq \bigcap \{\varphi \lor \psi\} (D)$$

We can then conclude similarly as in Lem. D.22 (Lem. F.9). Let $x \in \bigcap \{\varphi \lor \psi\} (D)$ and assume toward a contradiction that there are $S, S' \in D$ such that $x \notin \{\varphi\} (S)$ and $x \notin \{\psi\} (S')$. Then since $D$ is directed, there is some $S'' \in D$ such that $S \cup S' \subseteq S''$, and by antimonotonicity we get $x \notin \{\varphi \lor \psi\} (S'')$, a contradiction.

Case of $\psi \Rightarrow \varphi$.

With the classical semantics, the interpretation of $\Rightarrow$ can be decomposed into $\lor$ and $\land$, where $\{\neg \varphi\}$ is the complement of $\{\varphi\}$ (at the appropriate type). Let $\alpha$ be positive in $\varphi$ and negative in $\psi$, with $\alpha : Q^+ \vdash \varphi, \psi : P^+$.
Lemma F.13 (Monotonicity of Realizability (Lem. D.25)). Let $T$ be a type without free iteration variables. If $x \equiv_T T$ then $x \equiv_k T$ for all $k \leq n$.

Proof. By induction on the definition of $\equiv$. 

and let furthermore by $D$ and $D'$ (of the appropriate type) be resp. directed and codirected. We then trivially have

$$\{ \lnot \varphi \} (\bigcup D) = \mathcal{P}(\Gamma[[P^+]]) \setminus \{ \varphi \} (\bigcup D) = \mathcal{P}(\Gamma[[P^+]]) \setminus \{ \varphi \} (\bigcap D')$$

These modalities are handled similarly as in Lem. D.22 (Lem. F.9).

Case of $[\pi_i] \varphi$, $[\neg \pi_i] \varphi$, $[\text{next}] \varphi$ and $[\text{fold}] \varphi$.

Since $[\operatorname{ev}(\psi)] \varphi$ is smooth, the formula $\psi$ is closed and we have $Q^+ = B \rightarrow R^+$ with $B$ a finite base type. Since $B$ is constant, by Lem. D.4 there is a finite set $A$ such that $[B] \simeq \Delta A$, so that $\Gamma[[B]] \simeq A$ by Lem. D.2. Now, given $x \in \Gamma[[P^+]]$ and $S \subseteq \Gamma[[Q^+]]$ we have

$$x \in \{ [\operatorname{ev}(\psi)] \varphi \} (S) \text{ iff } \forall y \in A (y \in [\psi] \implies \operatorname{ev} \circ \langle x, y \rangle \in \{ [\varphi] \} (S))$$

Since $A$ is finite, we can then reason similarly as in the cases of conjunction ($\wedge$) above.

Cases of $\theta^+ \beta \varphi$.

We have $\alpha_1 : P_1^+, \ldots, \alpha_k : P_k^+, \alpha : Q^+, \beta : P^+ \vdash \varphi : P^+$ with $\beta$ Pos $\varphi$. Let $v$ be a valuation. Since for $S \subseteq \Gamma[[Q^+]]$ and $m \in \mathbb{N}$ we have

$$\{ [\theta^{m+1} \beta \varphi] \}_v (S) = \{ [\varphi[\theta^m \beta \varphi/\beta]] \}_v (S)$$

it follows from Lem. D.22 (Lem. F.9), that the function $\{ [\theta^m \beta \varphi] \}_v$, is monotone (resp. antimonotone) if $\alpha$ Pos $\varphi$ (resp. $\alpha$ Neg $\varphi$). We can then reason as in Lem. D.22 (Lem. F.9).

Case of $\nu \beta \varphi$.

We have $\alpha_1 : P_1^+, \ldots, \alpha_k : P_k^+, \alpha : Q^+, \beta : P^+ \vdash \varphi : P^+$ where the involved variables are gfp variables and are positive in $\varphi$. The result is then proved exactly the same way as in Lem. D.22 (Lem. F.9).

Case of $\mu \beta \varphi$.

The result is proved the same way as in Lem. F.9 (replacing codirected meets by directed joins and Scott cocontinuity by Scott continuity). 

F.4 Realizability
Case of a refinement type \{ A \mid \varphi \}.

The result follows from monotony of forcing (i.e. that \[\varphi\] is a subobject of \[A\])

Case of 1.

The result is trivial as \( x \models^\ast_n 1 \) for all \( n > 0 \).

Case of \( T_0 + T_1 \).

Assume \( x \models^\ast_n T_0 + T_1 \) and let \( k \leq n \). Then we have \( x = i_0 \circ y \) for some \( i = 0, 1 \) and some \( y \in \Gamma[T_i] \) such that \( y \models^\ast_n T_i \). By induction hypothesis we get \( y \models^\ast_k T_i \), so that \( x \models^\ast_k T_0 + T_1 \).

Case of \( T_0 \times T_1 \).

Assume \( x \models^\ast_n T_0 \times T_1 \) and let \( k \leq n \). Then for each \( i = 0, 1 \) we have \( \pi_i \circ x \models^\ast_n T_i \), so that \( \pi_i \circ x \models^\ast_k T_i \) by induction hypothesis, and it follows that \( x \models^\ast_k T_0 \times T_1 \).

Case of \( U \rightarrow T \).

Assume \( x \models^\ast_n U \rightarrow T \) and let \( k \leq n \). But given \( \ell \leq k \) and \( y \in \Gamma[U] \) such that \( y \models^\ast_\ell U \) we have \( \text{ev} \circ (x, y) \models^\ast_\ell T \) since \( \ell \leq n \).

Case of \( \text{Fix}(X).A \).

Assume \( x \models^\ast_n \text{Fix}(X).A \) and let \( k \leq n \). We have \( \text{unfold} \circ x \models^\ast_n A[\text{Fix}(X).A/X] \), so that \( \text{unfold} \circ x \models^\ast_k A[\text{Fix}(X).A/X] \) by induction hypothesis and thus \( x \models^\ast_k \text{Fix}(X).A \).

Case of \( \square T \).

Trivial.

□

Lemma F.14 (Lem., D.26). For a pure type \( A \) and \( x \in \Gamma[A] \), we have \( x \models^\ast_n A \) for all \( n > 0 \).

Proof. The proof is by induction on pairs \((n, A)\), using implicitly Lem., D.2 whenever required.

Case of 1.

Trivial.

Case of \( A_0 + A_1 \).

Given \( x \in \Gamma[A_0 + A_1] \simeq \Gamma[A_0] + \Gamma[A_1] \), we have \( x = i_0 \circ y \) for some \( y \in \Gamma[A_i] \). Then we are done since \( y \models^\ast_n A_i \) by induction hypothesis.

Case of \( A_0 \times A_1 \).

Given \( x \in \Gamma[A_0 \times A_1] \simeq \Gamma[A_0] \times \Gamma[A_1] \), we have \( \pi_0 \circ x \models^\ast_n A_0 \) and \( \pi_1 \circ x \models^\ast_n A_1 \) by induction hypothesis, and the result follows.

Case of \( B \rightarrow A \).

Fix \( x \in \Gamma[B \rightarrow A] \). Given \( y \in \Gamma[B] \) and \( k \leq n \), we have \( y \models^\ast_k B \) by induction hypothesis, so that \( \text{ev} \circ (x, y) \models^\ast_k A \). Hence \( x \models^\ast_n B \rightarrow A \).

Case of \( \square A \).

The result is trivial if \( n = 1 \), so assume \( n > 1 \). Given \( x \in \Gamma[\square A] \), we have
\( x = \text{next} \circ y \) for some \( y \in I[A] \). But then \( y \models_n A \) by induction hypothesis, so that \( x \models_n A \).

**Case of \( \text{Fix}(X).A \).**

Let \( x \in I[\text{Fix}(X).A] \). It follows by induction on \( A \) from the induction hypothesis on \( n \) and the guardedness of \( X \) in \( A \) that \( \text{unfold}\circ x \models_{n-1} A[\text{Fix}(X).A/X] \), and we are done.

**Case of \( \lnot T \).**

Let \( x \in I[\lnot T] \). Given \( n > 0 \), we have \( x_n(\bullet) \in I[T] \), so that \( x_n(\bullet) \models_m T \) for all \( m > 0 \) by induction hypothesis. But this implies \( x \models_n \lnot T \). \( \Box \)

**Lemma F.15 (Correctness of Subtyping (Lem. [D.28])).** Given types \( T,U \) without free iteration variable, if \( x \models_n U \) and \( U \preceq T \) then \( x \models_n T \).

**Proof.** By induction on \( U \preceq T \).

**Cases of**

\[
\begin{array}{ccc}
T \leq T & T \leq U & U \leq V \\
T \leq V & & \\
\end{array}
\]

**Trivial.**

**Cases of**

\[
\begin{array}{cccc}
T_0 \leq U_0 & T_1 \leq U_1 & T_0 \leq U_0 & T_1 \leq U_1 \\
T_0 \times T_1 \leq U_0 \times U_1 & T_0 + T_1 \leq U_0 + U_1 & U_0 \leq T_0 & T_1 \leq U_1 \\
T_0 \rightarrow T_1 \leq U_0 \rightarrow U_1 & & & \\
\end{array}
\]

**Trivial**

**Case of**

\[ U \leq T \]

\[ \lnot U \leq \lnot T \]

Let \( x : 1 \rightarrow_{\mathcal{S}} \Delta I[\lnot U] \) such that \( x \models_n \lnot U \), so that \( x_n(\bullet) \models_m U \) for all \( m > 0 \). By induction hypothesis we get \( x_n(\bullet) \models_m T \) for all \( m > 0 \) and we are done.

**Case of**

\[ T \leq [T] \]

By Lem. [D.26]

**Case of**

\[ A \leq \{A \mid \top\} \]

**Trivial**

**Case of**

\[ \vdash^A \varphi \Rightarrow \psi \]

\[ \{A \mid \varphi\} \leq \{A \mid \psi\} \]

By Lem. [F.5 (Lem. [D.13(2)]).]
Case of
\[
\{B \rightarrow A \mid [\text{ev}(\psi)] \varphi\} \leq \{B \mid \psi\} \rightarrow \{A \mid \varphi\}
\]

Let \( x \in \Gamma[B \rightarrow A] \) and \( n > 0 \). Assume \( x \parallel^\psi_n \{B \rightarrow A \mid [\text{ev}(\psi)] \varphi\} \), that is \( x_n(\bullet) \in [[\text{ev}(\psi)] \varphi](n) \). Let further \( y \in \Gamma[B] \) and \( k \leq n \) such that \( y \parallel_k \{B \mid \psi\} \), that is \( y_k(\bullet) \in [\psi](k) \). Then by monotonicity of \([-]\) (Lem. D.16) we have \( x_k(\bullet) \in [[\text{ev}(\psi)] \varphi](k) \), from which it follows that \((x_k(\bullet))(y_k(\bullet)) \in [\varphi](k) \). But this means \( \text{ev} \circ (x, y) \parallel_k \{A \mid \varphi\} \) and we are done.

Case of
\[
\{B \mid \psi\} \rightarrow \{A \mid \varphi\} \leq \{B \rightarrow A \mid [\text{ev}(\psi)] \varphi\}
\]

Let \( x \in \Gamma[B] \rightarrow A \) and \( n > 0 \). Assume \( x \parallel^\psi_n \{B \mid \psi\} \rightarrow \{A \mid \varphi\} \). Let furthermore \( k \leq n \) and \( u \in [\psi](k) \). By Lem. D.27 (Cor. 3.8) there is some \( y \in \Gamma[B] \) such that \( y_k(\bullet) = u \). We thus have \( y \parallel_k \{B \mid \psi\} \), and it follows that \( \text{ev} \circ (x, y) \parallel_k \{A \mid \varphi\} \), that is \( x_k(\bullet)(y_k(\bullet)) \in [\varphi](k) \), and we are done.

Case of
\[
\begin{array}{c}
\text{next} \{A \mid \varphi\} \\
\equiv \{\text{next} A \mid \varphi\}
\end{array}
\]

Let \( x \in \Gamma[\text{next} A] \). First, we always have \( x \parallel_{n-1} \text{next} A \), as well as \( x_1 \in [[\text{next} \varphi]] A \).

Let now \( n > 1 \). By Lem. D.2 we have \( x = \text{next} \circ y \) for some \( y \in \Gamma[A] \). Since \( x_n(\bullet) = y_{n-1}(\bullet) \), we have
\[
x \parallel_n \text{next} A \mid \varphi
\]

if \( y \parallel_{n-1} \{A \mid \varphi\} \)

if \( y_{n-1}(\bullet) \in [\varphi]A(n-1) \)

if \( x_n(\bullet) = y_{n-1}(\bullet) \in [[\text{next} \varphi]] A(n) \)

If \( x \parallel_n \{A \mid \text{next} \varphi\} \).

Case of
\[
\forall k \cdot \text{next} T \equiv \text{next} k \cdot T
\]

Let \( x \in \Gamma[\text{next} T] \).

Assume first that \( x \parallel_n \forall k \cdot \text{next} T \). We have to show \( x \parallel_n \forall k \cdot \text{next} T \). The result is trivial if \( n = 1 \). So assume \( n > 1 \). By Lem. D.2 there some unique \( y \in \Gamma[[T]] \) such that \( x = \text{next} \circ y \). We have to show \( y \parallel_{n-1} T[m/k] \) for all \( m \in \mathbb{N} \). But by assumption we have \( x \parallel_n \text{next} T[m/k] \), so that by uniqueness of \( y \) we get \( y \parallel_{n-1} T[m/k] \).

Conversely, assume that \( x \parallel_n \forall k \cdot \text{next} T \). We have to show \( x \parallel_n \forall k \cdot \text{next} T \).

Let \( m \in \mathbb{N} \). If \( n = 1 \), then we trivially have \( x \parallel_n \forall k \cdot \text{next} T[m/k] \). Otherwise, by Lem. D.2 let \( y \in \Gamma[[T]] \) such that \( x = \text{next} \circ y \). But since \( x \parallel_n \forall k \cdot T \), we get \( y \parallel_{n-1} T[m/k] \), so that \( x \parallel_n \forall k \cdot T[m/k] \) and we are done.

Case of
\[
\begin{array}{c}
\text{safe} \\
\text{box} \{A \mid \varphi\} \equiv \text{box} A \mid \text{box} \varphi
\end{array}
\]
Let $x : 1 \to S \Delta \Gamma[A]$. Since $\varphi$ is safe we have $\{\varphi\}^A = \text{Clos}([\varphi]_A)$ by Prop. 7.10 (Prop. 7.3). Then we are done since:

\[
\begin{align*}
x \vdash_n \llbracket \Gamma \mid \varphi \rrbracket & \iff \text{if } x_n(\bullet) \vdash_m \llbracket \Gamma \mid \varphi \rrbracket \text{ for all } m > 0 \\
& \text{if } (x_n(\bullet))_m(\bullet) \in [\varphi]^A(m) \text{ for all } m > 0 \\
& \text{if } x_n(\bullet) \in [\varphi]^A \\
& \text{if } x_n(\bullet) \in [\llbracket \text{box} \rrbracket]_A(n) \\
& \text{if } x \vdash_n \llbracket \Gamma \mid \llbracket \text{box} \rrbracket \rrbracket
\end{align*}
\]

Case of

\[
\vdash^A_c \varphi \Rightarrow \psi
\]

By Lem. F.2 (Lem. D.13 [1]).

Theorem F.16 (Adequacy (Thm. D.29)). Let $\mathcal{E}, T$ have free iteration variables among $\ell$, and let $\overline{m} \in \mathbb{N}$. If $\mathcal{E} \vdash M : T$ and $\rho \models \mathcal{E}$, then

\[
\forall n > 0, \quad \rho \vdash_n \mathcal{E}[\ell/\overline{m}] \implies \llbracket M \rrbracket_\rho \vdash_n T[\ell/\overline{m}]
\]

Proof. The proof is by induction on typing derivations. We implicitly use Lem. D.2 whenever required. We omit iteration variables when possible.

Case of

\[
\mathcal{E}, x : \triangleright T \vdash M : T \\
\mathcal{E} \vdash \text{fix}(x).M : T
\]

Let $\rho \models \mathcal{E}$ and write $y := [\text{fix}(x).M]_\rho \in \Gamma[T]$. Note that

\[
y = \llbracket M[\text{next}(\text{fix}(x).M)/x] \rrbracket_\rho = \llbracket M[\text{next}_\rho/y] \rrbracket
\]

We show by induction on $n > 0$ that $\rho \vdash_n \mathcal{E}$ implies $y \vdash_n T$. In the base case $n = 1$, since $\text{next} \circ y \vdash_1 \triangleright T$, we have $\rho[\text{next} \circ y/x] \vdash_1 \mathcal{E}, x : \triangleright T$, so that the induction hypothesis on typing derivations gives $y = \llbracket M[\text{next}_\rho/y] \rrbracket$. As for induction step, assume $\rho \vdash_{n+1} \mathcal{E}$. By Monotonicity of Realizability (Lem. F.13), we have $\rho \vdash_n \mathcal{E}$, and the induction hypothesis on $n$ gives $y \vdash_n T$. It follows that $\text{next} \circ y \vdash_{n+1} \triangleright T$, so that $\rho[\text{next} \circ y/x] \vdash_{n+1} \mathcal{E}, x : \triangleright T$ and the induction hypothesis on typing derivations gives $y = \llbracket M[\text{next}_\rho/y] \rrbracket$. In case of

\[
\mathcal{E} \vdash M : T \\
\mathcal{E} \vdash \text{next}(M) : \triangleright T
\]

Let $\rho \models \mathcal{E}$ and write $y := [\text{next}(M)]_\rho \in \Gamma[\triangleright T]$. Let $n > 0$ such that $\rho \vdash_n T$. If $n = 1$ then we trivially have $x \vdash_1 \triangleright T$. Assume $n > 1$. Write $y := [M]_\rho$, so that $x = \text{next} \circ y$. By Monotonicity of Realizability (Lem. F.13), we have $\rho \vdash_{n-1} \mathcal{E}$, so that the induction hypothesis on typing derivations gives $y \vdash_{n-1} T$ and we are done.
Case of
\[
x_1 : T_1, \ldots, x_k : T_k \vdash M : T \quad \mathcal{E} \vdash M_1 : T_1 \quad \ldots \quad \mathcal{E} \vdash M_k : T_k \\
\mathcal{E} \vdash \operatorname{box}_{[x_1 \mapsto M_1, \ldots, x_k \mapsto M_k]}(M) : \mathbf{■}T
\]

Let \( \rho \models \mathcal{E} \) and write \( x := [\operatorname{box}_\rho(M)]_\rho \), where \( \sigma = [x_1 \mapsto M_1, \ldots, x_k \mapsto M_k] \).

Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). We show \( x \models_n \mathbf{■}T \), i.e. that \( x_m(\bullet) \models_m T \) for all \( m > 0 \).

Fix \( m > 0 \). We have by definition
\[
x_m(\bullet) : \ell \mapsto [M]_\ell(\langle [M_1]_m(\rho_m(\bullet)), \ldots, [M_k]_m(\rho_m(\bullet)) \rangle)
\]

For \( i = 1, \ldots, k \), since the type \( T_i \) is constant, we have by Lem. D.24 that 
\[
[M_i]_m(\rho_m(\bullet)) = [M_i]_\ell(\rho_i(\bullet)) \quad \text{for all} \quad \ell > 0,
\]
so that
\[
x_m(\bullet) = \ell \mapsto [M]_\ell(\langle [M_1]_\ell(\rho_1(\bullet)), \ldots, [M_k]_\ell(\rho_k(\bullet)) \rangle)
\]

Now, by induction hypothesis, since \( \rho \models_n \mathcal{E} \) by assumption, for each \( i = 1, \ldots, k \) we have \( [M_i]_\rho \models_n T_i \) and since \( T_i \) is constant, by Lem. D.24 this implies \( [M_i]_\rho \models_\ell T_i \) for all \( \ell > 0 \). By induction hypothesis again, this in turn gives \( [M]_\rho \circ (\langle [M_1]_\rho, \ldots, [M_k]_\rho \rangle) \models_\ell T \) for each \( \ell > 0 \). But then we are done since
\[
x_m(\bullet) = \ell \mapsto [M]_\ell(\langle [M_1]_\rho(\rho_1(\bullet)), \ldots, [M_k]_\rho(\rho_k(\bullet)) \rangle)
\]

Case of
\[
\mathcal{E} \vdash M : \mathbf{■}T
\]
\[
\mathcal{E} \vdash \operatorname{unbox}(M) : T
\]

Let \( \rho \models \mathcal{E} \) and write \( x := [\operatorname{unbox}(M)]_\rho \). Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). By induction hypothesis we get \( [M]_\rho \models_n \mathbf{■}T \), that is \( ([M]_\rho)_m(\bullet) \models_m T \) for all \( m > 0 \), so in particular \( ([M]_\rho)_n(\bullet) \models_n T \). But now we are done since 
\[
x_m(\bullet) = ([M]_\rho)_n(\bullet)_m(\bullet) \quad \text{for each} \quad m > 0.
\]

Case of
\[
x_1 : T_1, \ldots, x_k : T_k \vdash M : \mathbf{◆}T \quad \mathcal{E} \vdash M_1 : T_1 \quad \ldots \quad \mathcal{E} \vdash M_k : T_k \\
\mathcal{E} \vdash \operatorname{prev}_{[x_1 \mapsto M_1, \ldots, x_k \mapsto M_k]}(M) : \mathbf{■}T
\]

Let \( \rho \models \mathcal{E} \) and write \( x := [\operatorname{box}_\rho(M)]_\rho \), where \( \sigma = [x_1 \mapsto M_1, \ldots, x_k \mapsto M_k] \).

Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). We show \( x \models_n \mathbf{◆}T \). If \( n = 1 \) then the result trivially holds. Assume \( n > 1 \). For each \( m > 0 \), we have by definition
\[
x_m(\bullet) = [M]_{m+1}(\langle [M_1]_m(\rho_m(\bullet)), \ldots, [M_k]_m(\rho_m(\bullet)) \rangle)
\]

For \( i = 1, \ldots, k \), since the type \( T_i \) is constant, we have by Lem. D.24 that 
\[
[M_i]_m(\rho_m(\bullet)) = [M_i]_{m+1}(\rho_{m+1}(\bullet)),
\]
so that
\[
x_m(\bullet) = [M]_{m+1}(\langle [M_1]_{m+1}(\rho_{m+1}(\bullet)), \ldots, [M_k]_{m+1}(\rho_{m+1}(\bullet)) \rangle)
\]
and it follows that
\[
x = \text{next} \circ [M] \circ ([M_1], \ldots, [M_k], x)
\]

Now, by induction hypothesis, since \( \rho \parallel n \mathcal{E} \) by assumption, for each \( i = 1, \ldots, k \) we have \([M_i], \rho \parallel n T_i\) and since \( T_i \) is constant, by Lem. \[D.24\] this implies \([M_i], \rho \parallel n-1 T_i\). By induction hypothesis again, this in turn gives
\([M], \rho \parallel (\[M_1], \ldots, [M_k], x) \parallel n-1 T\) and we are done.

Case of
\[
\mathcal{E} \vdash M : T \quad T \leq U
\]

By Lem. \[D.28\] (Lem. \([F.15]\),

Case of
\[
\mathcal{E} \vdash M : \{A \mid \psi \Rightarrow \varphi\} \quad \mathcal{E} \vdash M : \{A \mid \psi\}
\]

Let \( \rho \models \mathcal{E} \) and write \( x := [M], \rho \models \Gamma[A] \). Let \( n > 0 \) such that \( \rho \parallel n \mathcal{E} \). By induction hypothesis, the right premise gives \( x_n(\bullet) \in \llbracket \psi \rrbracket^A(n) \) and the left premise implies \( x_n(\bullet) \in \llbracket \varphi \rrbracket^A(n) \).

Case of
\[
\mathcal{E} \vdash M : \{A \mid \varphi_0 \vee \varphi_1\} \quad \mathcal{E}, x : \{A \mid \varphi_1\} \vdash N : U
\]

Let \( \rho \models \mathcal{E} \) and write \( x := [M], \rho \models \Gamma[A] \) and \( z := [N], \rho[y/x] \models \Gamma[U] \). Let \( n > 0 \) and assume \( \rho \parallel n \mathcal{E} \). By induction hypothesis we have \( y \in \llbracket \varphi_i \rrbracket \) for some \( i \in \{0, 1\} \). It follows that \( \rho[y/x] \parallel n \mathcal{E}, x : \{A \mid \varphi_i\} \), from which we get \( z \parallel n B \) by induction hypothesis.

Case of
\[
\mathcal{E} \vdash M : \{A \mid \bot\} \quad \mathcal{E} \vdash N : [U]
\]

Let \( \rho \models \mathcal{E} \) and write \( x := [M], \rho \models \Gamma[A] \). Let \( n > 0 \) such that \( \rho \parallel n \mathcal{E} \). By induction hypothesis, the left premise gives \( x_n(\bullet) \in [\bot]^A(n) = \emptyset \), a contradiction. Hence \( \rho \not\parallel n \mathcal{E} \), and the result follows.

Case of
\[
\mathcal{E} \vdash M_i : \{A_i \mid \varphi\} \quad \mathcal{E} \vdash M_{i+1} : A_{i+1}
\]

Let \( \rho \models \mathcal{E} \). Write \( y_0 := [M_0], \rho \models \Gamma[A_0], y_1 := [M_1], \rho \models \Gamma[A_1], \) and \( x := ([M_0, M_1], \rho) = (y_0, y_1) \). Let \( n > 0 \) such that \( \rho \parallel n \mathcal{E} \). By induction hypothesis on typing derivations we have \( (y_i)_n(\bullet) \in \llbracket \varphi \rrbracket \). But since \( \pi_i(x_n(\bullet)) = (y_i)_n(\bullet) \), this gives \( x_n(\bullet) \in \llbracket \pi_i[\varphi] \rrbracket \).

Case of
\[
\mathcal{E} \vdash M : \{A_0 \times A_1 \mid [\pi_i][\varphi]\}
\]

Let \( \rho \models \mathcal{E} \). Write \( y := [M], \rho \models \Gamma[A_0 \times A_1] \) and \( x := [\pi_i(M)], \rho = \pi_i \circ y \). Let \( n > 0 \) such that \( \rho \parallel n \mathcal{E} \). By induction hypothesis on typing derivations we have \( y_n(\bullet) \in [\pi_i[\varphi]] \), so that \( \pi_i(y_n(\bullet)) \in [\varphi] \). But then we are done since \( x_n(\bullet) = \pi_i(y_n(\bullet)) \).
Case of
\[ \mathcal{E} \vdash M : \{ A_1 \mid \varphi \} \]
\[ \mathcal{E} \vdash \text{in}_i(M) : \{ A_0 + A_1 \mid [i_n] \varphi \} \]

Let \( \rho \models \mathcal{E} \). Write \( y := [M]_\rho \in \Gamma[A_1] \), and \( x := [\text{in}_i(M)]_\rho = \text{in}_i \circ y \). Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). Hence by induction hypothesis on typing derivations we have \( y_n(\bullet) \in [\varphi] \). But since \( x_n(\bullet) = \text{in}_i(y_n(\bullet)) \), this implies \( x_n(\bullet) \in [i_n] \varphi \).

Case of
\[ \mathcal{E} \vdash M : \{ A_0 + A_1 \mid [i_n] \varphi \} \]
\[ \mathcal{E}, x : \{ A_i \mid \varphi \} \vdash N_i : U \]
\[ \mathcal{E}, x : A_{i-i} \vdash N_{i-i} : U \]
\[ \mathcal{E} \vdash \text{case} \text{ of } (x.N_0|x.N_1) : U \]

Let \( \rho \models \mathcal{E} \). Write \( y := [M]_\rho \in \Gamma[A_0 + A_1] \cong \Gamma[A_0] + \Gamma[A_1] \). Hence \( y = \text{in}_i \circ z \) for some (unique) \( j \in \{0, 1\} \) and \( z \in \Gamma[A_j] \). Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). By induction hypothesis, the left premise gives \( y_n(\bullet) \in [\varphi](n) \), so that \( y_n(\bullet) = \text{in}_i(u) \) for some \( u \in [\varphi](n) \). But this implies \( j = i \) and \( u = z_n(\bullet) \), so that \( z \models_n \mathcal{E}, x : \{ A_i \mid \varphi \} \). It follows that \( \rho[z/x] \models_n \mathcal{E} \vdash x : \{ A_i \mid \varphi \} \), and the induction hypothesis on typing derivations gives \( [N_i]_{\rho[z/x]} \models_n U \).

But then we are done since
\[ [\text{case } M \text{ of } (x.N_0|x.N_1)]_\rho = [N_i]_{\rho[z/x]} \]

Case of
\[ \mathcal{E}, x : \{ B \mid \psi \} \vdash M : \{ A \mid \varphi \} \]
\[ \mathcal{E} \vdash \lambda x. M : \{ B \rightarrow A \mid \text{ev} (\psi) \} \]

Let \( \rho \models \mathcal{E} \). Write \( y := [\lambda x. M]_\rho \in \Gamma[B \rightarrow A] \). Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). We show \( y_n(\bullet) \in [\text{ev}(\psi)](n) \).

By [20, Cor. 3.8] there is some \( z \in \Gamma[B] \) such that \( z_k(\bullet) = t \). By Monotonicity of Realizability (Lem. [21,13], we have \( \rho \models_k \mathcal{E} \), so that \( \rho[z/x] \models_k \mathcal{E}, x : \{ B \mid \psi \} \). The induction hypothesis on typing derivations thus gives \( ([M]_{\rho[z/x]})(k(\bullet)) \in [\varphi] \), and we are done since \( (y_n(\bullet))(z_k(\bullet)) = ([M]_{\rho[z/x]})(k(\bullet)) \).

Case of
\[ \mathcal{E} \vdash M : \{ B \rightarrow A \mid \text{ev}(\psi) \} \]
\[ \mathcal{E} \vdash N : \{ B \mid \psi \} \]
\[ \mathcal{E} \vdash MN : \{ A \mid \varphi \} \]

Let \( \rho \models \mathcal{E} \). Write \( y := [M]_\rho \in \Gamma[B \rightarrow A] \), \( z := [N]_\rho \in \Gamma[B] \) and \( x := [MN]_\rho \) = \text{ev} \circ (y,z). \)\ Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). By induction on typing derivations, the right premise gives \( z_n(\bullet) \in [\psi](n) \), so that the left premise gives \( (y_n(\bullet))(z_n(\bullet)) \in [\varphi](n) \). But then we are done since \( x_n(\bullet) = (y_n(\bullet))(z_n(\bullet)) \).

Case of
\[ \mathcal{E} \vdash M : \{ A[\text{Fix}(X),A/X] \mid \varphi \} \]
\[ \mathcal{E} \vdash \text{fold}(M) : \{ \text{Fix}(X),A \mid [\text{fold}] \} \]

Let \( \rho \models \mathcal{E} \). Write \( y := [M]_\rho \in \Gamma[A[\text{Fix}(X),A/X]] \) and \( x := [\text{fold}(M)]_\rho = \text{fold} \circ y \). \)\ Let \( n > 0 \) such that \( \rho \models_n \mathcal{E} \). By induction hypothesis on typing derivations we have \( y_n(\bullet) \in [\varphi] \). But then we are done since \( \text{unfold}_n(x_n(\bullet)) = y_n(\bullet) \).
Cases of

\( E \vdash M : \{ \text{Fix}(X).A \mid [\text{fold}]_\varphi \} \)

\( E \vdash \text{unfold}(M) : \{ A[\text{Fix}(X).A/X] \mid \varphi \} \)

Let \( \rho \models E \). Write \( y := [M]_\rho \in T[\text{Fix}(X).A] \) and \( x := [\text{unfold}(M)]_\rho = \text{unfold} \circ y \). Let \( n > 0 \) such that \( \rho \models_n E \). By induction hypothesis on typing derivations we have \( y_n(\bullet) \in [[\text{fold}]_\varphi \varphi] \). Hence \( \text{unfold}_n(y_n(\bullet)) \in [\varphi] \) and we are done since \( x_n(\bullet) = \text{unfold}_n(y_n(\bullet)) \).

Cases of

\[
\begin{align*}
E & \vdash M : T[0/\ell] & E & \vdash M : T[\ell+1/\ell] & (\ell \text{ not free in } E) \\
E & \vdash M : \forall \ell : T & E & \vdash M : \forall \ell : T & (\ell \text{ not free in } E)
\end{align*}
\]

Let \( \rho \models E \) and write \( x := [M]_\rho \in T[\ell] \). Let \( n > 0 \) and assume \( \rho \models_n E \). Let \( m \in \mathbb{N} \). We have to show \( M \models_n T[m/\ell] \). Since \( \ell \) does not occur free in \( E \), we have \( \rho \models_n E[m'/\ell] \) for all \( m' \in \mathbb{N} \). For both rules we can conclude from the induction hypothesis.

Case of

\[ E \vdash M : \forall \ell : T \]

\[ E \vdash M : T[t/\ell] \]

Let \( \rho \models E \) and write \( x := [M]_\rho \in T[\ell] \). Let \( n > 0 \) and assume \( \rho \models_n E \).

By induction hypothesis we have \( x \models_n T[m/\ell] \) for \( m = [t] \) and the result follows.

Cases of

\[
\begin{align*}
E & \vdash M : \{ \mathbf{A} \mid [\boxdot]_{\gamma} [\nu^\ell \alpha \varphi/\beta] \} & & \beta \text{ Pos } \gamma \\
E & \vdash M : \{ \mathbf{A} \mid [\boxdot]_{\gamma} [\nu \alpha \varphi/\beta] \}
\end{align*}
\]

\[ E \vdash M : \{ \mathbf{A} \mid [\boxdot]_{\gamma} [\mu^\ell \alpha \varphi/\beta] \} \]

\[ E, x : \{ \mathbf{A} \mid [\boxdot]_{\gamma} [\mu^\ell \alpha \varphi/\beta] \} \vdash N : U \]

\[ \beta \text{ Pos } \gamma \\
E \vdash N[M/x] : U \]

where \( \ell \) is not free in \( E, U, \gamma \), and \( \gamma, \varphi \) are smooth. First, since \( \varphi \) is smooth by Lem. \( \ref{lem:smoothing} \), we have

\[
\{ [\nu \alpha \varphi(\alpha)] \} = \bigcap_{m \in \mathbb{N}} \{ [\varphi^m(T)] \}
\]

and

\[
\{ [\mu \alpha \varphi(\alpha)] \} = \bigcup_{m \in \mathbb{N}} \{ [\varphi^m(T)] \}
\]

Moreover, since \( \beta \) is positive in \( \gamma \) and \( \gamma \) is smooth, it follows from Lem. \( \ref{lem:smoothing} \) (Lem. \( \ref{lem:smoothing} \)) that \( \{ [\gamma] \} \) is continuous and cocontinuous in \( \beta \). We thus get

\[
\{ [\gamma [\nu \alpha \varphi(\alpha)]/\beta] \} = \bigcap_{m \in \mathbb{N}} \{ [\gamma [\varphi^m(T)]/\beta] \}
\]

and

\[
\{ [\gamma [\mu \alpha \varphi(\alpha)]/\beta] \} = \bigcup_{m \in \mathbb{N}} \{ [\gamma [\varphi^m(T)]/\beta] \}
\]

and the result follows. \( \square \)
Table of Contents

Temporal Refinements for Guarded Recursive Types

Guilhem Jaber and Colin Riba

1 Introduction ............................................. 1
   Organization of the paper............................... 3
2 Outline ................................................ 3
   Overview of the Guarded λ-Calculus ................... 3
   Compositional Safety Reasoning on Streams .......... 3
   A Many-sorted Temporal Logic ......................... 4
   Beyond Safety ............................................ 5
   Internal Semantics in the Topos of Trees ............ 5
   The Necessity of an External Semantics .......... 5
   The Constant Type Modality ............................ 6
   Approximating Least Fixpoints ......................... 6
3 The Pure Calculus ....................................... 6
4 A Temporal Modal Logic ................................ 8
   Many-sorted Modal Temporal Formulae .......... 8
   Modal Theories ......................................... 10
5 A Temporally Refined Type System ..................... 11
6 The Full System ....................................... 12
   Strictly Positive and Polynomial Types .......... 13
   The Full Temporal Modal Logic ....................... 13
   The Safe and Smooth Fragments .................... 14
   The Full System ....................................... 15
7 Semantics ................................................. 16
   Denotational Semantics in the Topos of Trees ....... 16
   External Semantics ...................................... 17
   Internal Semantics of Formulae ...................... 17
   The Safe Fragment ....................................... 18
   The Smooth Fragment .................................... 19
   The Realizability Semantics ......................... 19
8 Examples .................................................. 19
9 Related Work ........................................... 23
10 Conclusion and Future Work ............................ 24
A Additional Material for §3 (The Pure Calculus) .... 32
B Additional Material for §4 (A Temporal Modal Logic) 32
C Additional Material for §5 (A Temporally Refined Type System) 33
D Additional Material for §7 (Semantics) ................ 34
   D.1 The Topos of Trees (Basic Structure) ............ 35
   D.2 Global Sections and Constant Objects ............ 35
   D.3 External and Internal Semantics: Global Definitions 37
   D.4 An Open Geometric Morphism ....................... 39