Commuting Rules for the Later Modality and Quantifiers in Step-Indexed Logics

Bálint Kocsis Robbert Krebbers

Radboud University

TYPES, June 11, 2025

Summary

Provide novel rules for commuting the later modality with quantifiers in step-indexed logics, sound with respect to their semantics in the topos of trees

Summary

Provide novel rules for commuting the later modality with quantifiers in step-indexed logics, sound with respect to their semantics in the topos of trees

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{ex} \circledast Q) \dashv\vdash \exists y : \blacktriangleright A.\, \texttt{lift}\,(Q \circledast y)}$$

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{all} \circledast Q) \dashv\vdash \forall y : \blacktriangleright A.\, \texttt{lift}(Q \circledast y)}$$

Summary

Provide novel rules for commuting the later modality with quantifiers in step-indexed logics, sound with respect to their semantics in the topos of trees

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{ex} \circledast Q) \dashv\vdash \exists y : \blacktriangleright A.\, \texttt{lift}\,(Q \circledast y)}$$

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{all} \circledast Q) \dashv\vdash \forall y : \blacktriangleright A.\, \texttt{lift}(Q \circledast y)}$$

Also, everything is formalised in Rocq ;)

Outline

Question

How do we ensure that a recursive definition is **well-defined**?

Question

How do we ensure that a recursive definition is **well-defined**?

Example

For $n \in \mathbb{N}$: $n! = 1 \cdot 2 \cdot \ldots \cdot n$.

Question

How do we ensure that a recursive definition is **well-defined**?

Example

For $n \in \mathbb{N}$: $n! = 1 \cdot 2 \cdot \ldots \cdot n$. Recursively:

$$n! = \begin{cases} 1 & \text{if n} = 0 \\ n \cdot (n-1)! & \text{if } n > 0 \end{cases}$$

Question

How do we ensure that a recursive definition is **well-defined**?

Example

For $n \in \mathbb{N}$: $n! = 1 \cdot 2 \cdot \ldots \cdot n$. Recursively:

$$n! = \begin{cases} 1 & \text{if n} = 0 \\ n \cdot (n-1)! & \text{if } n > 0 \end{cases}$$

Example

For $a \in \mathbb{N}^{\mathbb{N}}$: $f((a_n)_{n \in \mathbb{N}}) = (a_n + 1)_{n \in \mathbb{N}}$.

Question

How do we ensure that a recursive definition is **well-defined**?

Example

For $n \in \mathbb{N}$: $n! = 1 \cdot 2 \cdot \ldots \cdot n$. Recursively:

$$n! = \begin{cases} 1 & \text{if n} = 0 \\ n \cdot (n-1)! & \text{if } n > 0 \end{cases}$$

Example

For $a \in \mathbb{N}^{\mathbb{N}}$: $f((a_n)_{n \in \mathbb{N}}) = (a_n + 1)_{n \in \mathbb{N}}$. Recursively:

$$f(a) = (a_0 + 1, f(a \circ s)) \quad (\text{where } s(n) = n + 1)$$

Solution 1

Structural recursion on **inductive data types**

Solution 1

Structural recursion on **inductive data types**

Example

```
data N : Set where
  Z : N
  S : N -> N

_*_ : N -> N -> N
...

fact : N -> N
fact 0     = 1
fact (S n) = fact n * S n
```

Solution 1

Structural recursion on **inductive data types**

Example

```
data N : Set where
  Z : N
  S : N -> N

_*_ : N -> N -> N
...

fact : N -> N
fact 0     = 1
fact (S n) = fact n * S n
```

Encodes **well-founded** definitions

Solution 2

Structural corecursion on **coinductive data types**

Solution 2

Structural corecursion on **coinductive data types**

Example

```
record Str : Set where
  coinductive
  field
    hd : N
    tl : Str
open S

inc : Str -> Str
inc a .hd = S (a .hd)
inc a .tl = inc (a .tl)
```

Solution 2

Structural corecursion on **coinductive data types**

Example

```
record Str : Set where
  coinductive
  field
    hd : N
    tl : Str
open S

inc : Str -> Str
inc a .hd = S (a .hd)
inc a .tl = inc (a .tl)
```

Encodes **productive** definitions

Problem

- Inductive and coinductive types have to be **strictly positive** in order to guarantee termination (hence well-definedness)

Problem

- Inductive and coinductive types have to be **strictly positive** in order to guarantee termination (hence well-definedness)
- What about more exotic domains? E.g.

```
data Exp =
      Var String
    | App Exp Exp
    | Lam (Exp -> Exp)
```

Solution 3

- Key idea: step-wise **approximation**

Solution 3

- Key idea: step-wise **approximation**
- Step-indexing: semantic tool for stratifying recursive definitions [AM01, Ahm04]

Solution 3

- Key idea: step-wise **approximation**
- Step-indexing: semantic tool for stratifying recursive definitions [AM01, Ahm04]
- Approximation modality: modal framework for expressing self-referential formulas [Nak00]

Solution 3

- Key idea: step-wise **approximation**
- Step-indexing: semantic tool for stratifying recursive definitions [AM01, Ahm04]
- Approximation modality: modal framework for expressing self-referential formulas [Nak00]
- Marry the two [AMRV07] and coin the term “later modality”

Guarded type theory

- New type former $\blacktriangleright$ (pronounced "later")
  - Allows us to talk about data we will only have access to in the next computation step
  - Guards recursive occurrences in recursively defined types and terms

Guarded type theory

- New type former $\blacktriangleright$ (pronounced "later")
  - Allows us to talk about data we will only have access to in the next computation step
  - Guards recursive occurrences in recursively defined types and terms
- Applicative structure
  - $\texttt{next} : A \to \blacktriangleright A$: shifts data into the future
  - $- \circledast - : \blacktriangleright(A \to B) \to \blacktriangleright A \to \blacktriangleright B$: applies a function in the future

Guarded type theory

- New type former $\blacktriangleright$ (pronounced "later")
  - Allows us to talk about data we will only have access to in the next computation step
  - Guards recursive occurrences in recursively defined types and terms
- Applicative structure
  - $\mathtt{next} : A \to \blacktriangleright A$: shifts data into the future
  - $- \circledast - : \blacktriangleright(A \to B) \to \blacktriangleright A \to \blacktriangleright B$: applies a function in the future
- Guarded fixed point combinator: $\mathtt{fix} : (\blacktriangleright A \to A) \to A$
  - Self-reference delayed in time by $\mathtt{next}$:

$$\mathtt{fix}\, f = f\, (\mathtt{next}\, (\mathtt{fix}\, f))$$

  - We write $\mu x : \blacktriangleright A.\, t$ for $\mathtt{fix}\, (\lambda x : \blacktriangleright A.\, t)$

Motivating example: streams

- $\texttt{Str} = \mu X.\mathbb{N} \times \blacktriangleright X$, hence $\texttt{Str} = \mathbb{N} \times \blacktriangleright \texttt{Str}$

Motivating example: streams

- $\mathtt{Str} = \mu X.\mathbb{N} \times \blacktriangleright X$, hence $\mathtt{Str} = \mathbb{N} \times \blacktriangleright \mathtt{Str}$
- Constructors and destructors:

$$- :: - : \mathbb{N} \to \blacktriangleright \mathtt{Str} \to \mathtt{Str}$$

$$\mathtt{hd} : \mathtt{Str} \to \mathbb{N}$$

$$\mathtt{tl} : \mathtt{Str} \to \blacktriangleright \mathtt{Str}$$

Motivating example: streams

- $\texttt{Str} = \mu X. \mathbb{N} \times \blacktriangleright X$, hence $\texttt{Str} = \mathbb{N} \times \blacktriangleright \texttt{Str}$
- Constructors and destructors:

$$- :: - : \mathbb{N} \to \blacktriangleright \texttt{Str} \to \texttt{Str}$$
$$\texttt{hd} : \texttt{Str} \to \mathbb{N}$$
$$\texttt{tl} : \texttt{Str} \to \blacktriangleright \texttt{Str}$$

- Recursive operations:

$$\texttt{zeros} = \mu s : \blacktriangleright \texttt{Str}.\, 0 :: s$$
$$\texttt{inc} = \mu r : \blacktriangleright (\texttt{Str} \to \texttt{Str}).\, \lambda s : \texttt{Str}.\, (\texttt{hd}\, s + 1) :: (r \circledast \texttt{tl}\, s)$$

Step-indexed logic

Under the Curry-Howard correspondence, guarded recursive operations correspond to logical connectives and rules

Step-indexed logic

Under the Curry-Howard correspondence, guarded recursive operations correspond to logical connectives and rules

- ► type former $\Rightarrow \triangleright$ modality (also pronounced "later")
  - $\triangleright P$ holds now if and only if P holds at the next step

Step-indexed logic

Under the Curry-Howard correspondence, guarded recursive operations correspond to logical connectives and rules

- $\blacktriangleright$ type former $\Rightarrow$ $\triangleright$ modality (also pronounced "later")
  - $\triangleright P$ holds now if and only if P holds at the next step
- Applicative structure $\Rightarrow$ modal axioms:

$$P \vdash \triangleright P \qquad\qquad \triangleright(P \supset Q) \vdash \triangleright P \supset \triangleright Q$$

Step-indexed logic

Under the Curry-Howard correspondence, guarded recursive operations correspond to logical connectives and rules

- $\blacktriangleright$ type former $\Rightarrow$ $\triangleright$ modality (also pronounced "later")
  - $\triangleright P$ holds now if and only if P holds at the next step
- Applicative structure $\Rightarrow$ modal axioms:

$$P \vdash \triangleright P \qquad\qquad \triangleright(P \supset Q) \vdash \triangleright P \supset \triangleright Q$$

- Fixed point combinator $\Rightarrow$ Löb rule:

$$\triangleright P \supset P \vdash P$$

In short: to prove P, we can assume that P already holds after one computation step

Crucial rules

$$P \vdash \triangleright P \qquad \frac{P \vdash Q}{\triangleright P \vdash \triangleright Q} \qquad \triangleright P \supset P \vdash P$$

$$\triangleright (P * Q) \dashv\vdash \triangleright P * \triangleright Q \quad (* \in \{\wedge, \vee, \supset\}) \qquad \triangleright (t =_A u) \dashv\vdash \mathtt{next}\, t =_{\blacktriangleright A} \mathtt{next}\, u$$

Semantics

- Intuitively: sequences of approximations
  - The n-th element describes what the object looks like if one has only n steps to reason about it
  - n: step-index
  - $\blacktriangleright$ and $\triangleright$ shift step-indices

Semantics

- Intuitively: sequences of approximations
  - The n-th element describes what the object looks like if one has only n steps to reason about it
  - n: step-index
  - $\blacktriangleright$ and $\triangleright$ shift step-indices
- Two main formalisms:
  - Ordered families of equivalences (used by Iris [JKJ$^+$18])
  - Topos of trees

Semantics

- Intuitively: sequences of approximations
  - The n-th element describes what the object looks like if one has only n steps to reason about it
  - n: step-index
  - $\blacktriangleright$ and $\triangleright$ shift step-indices
- Two main formalisms:
  - Ordered families of equivalences (used by Iris [JKJ+18])
  - Topos of trees
- General models of guarded recursion [BMSS12], in particular sheaves over certain complete Heyting-algebras

Topos of trees

- $\mathcal{S}$: presheaves on the ordinal ω

Topos of trees

- $\mathcal{S}$: presheaves on the ordinal ω
- Objects X:

$$X_0 \xleftarrow{r_0^X} X_1 \xleftarrow{r_1^X} X_2 \xleftarrow{r_2^X} \cdots$$

Topos of trees

- $\mathcal{S}$: presheaves on the ordinal ω
- Objects X:

$$X_0 \xleftarrow{r_0^X} X_1 \xleftarrow{r_1^X} X_2 \xleftarrow{r_2^X} \cdots$$

Notation: $x|_n = (r_n^X \circ r_{n+1}^X \circ \ldots \circ r_{m-1}^X)(x)$

- Morphisms $f : X \to Y$:

$$\begin{array}{ccccccc} X_0 & \xleftarrow{r_0^X} & X_1 & \xleftarrow{r_1^X} & X_2 & \xleftarrow{r_2^X} & \cdots \\ {\scriptstyle f_0}\downarrow & & {\scriptstyle f_1}\downarrow & & {\scriptstyle f_2}\downarrow & & \\ Y_0 & \xleftarrow{r_0^Y} & Y_1 & \xleftarrow{r_1^Y} & Y_2 & \xleftarrow{r_2^Y} & \cdots \end{array}$$

Guarded recursion

- $\blacktriangleright : \mathcal{S} \to \mathcal{S}$ sends X to

$$\{*\} \xleftarrow{\ !\ } X_0 \xleftarrow{\ r_0\ } X_1 \xleftarrow{\ r_1\ } \cdots$$

Guarded recursion

- $\blacktriangleright : \mathcal{S} \to \mathcal{S}$ sends X to

$$\{*\} \xleftarrow{\;!\;} X_0 \xleftarrow{\;r_0\;} X_1 \xleftarrow{\;r_1\;} \cdots$$

- $\mathtt{next}_X : X \to \blacktriangleright X$, $(\mathtt{next}_X)_n = r_n^{\blacktriangleright X}$

$$\begin{array}{ccccccc}
X_0 & \xleftarrow{\;r_0^X\;} & X_1 & \xleftarrow{\;r_1^X\;} & X_2 & \xleftarrow{\;r_2^X\;} & \cdots \\
{\scriptstyle !}\downarrow & & {\scriptstyle r_0^X}\downarrow & & {\scriptstyle r_1^X}\downarrow & & \\
\{*\} & \xleftarrow[\;!\;]{} & X_0 & \xleftarrow[\;r_0^X\;]{} & X_1 & \xleftarrow[\;r_1^X\;]{} & \cdots
\end{array}$$

Example

- Streams:

$$\mathbb{N} \xleftarrow{\pi_1} \mathbb{N} \times \mathbb{N} \xleftarrow{\pi_1} \mathbb{N} \times \mathbb{N} \times \mathbb{N} \xleftarrow{\pi_1} \cdots$$

Example

- Streams:

$$\mathbb{N} \xleftarrow{\pi_1} \mathbb{N} \times \mathbb{N} \xleftarrow{\pi_1} \mathbb{N} \times \mathbb{N} \times \mathbb{N} \xleftarrow{\pi_1} \cdots$$

- $\mathtt{hd} \colon \mathtt{Str} \to \mathbb{N}$, $\mathtt{hd}_n(s_0, \ldots, s_n) = s_0$
- $\mathtt{inc} : \mathtt{Str} \to \mathtt{Str}$, $\mathtt{inc}_n(s_0, \ldots, s_n) = (s_0 + 1, \ldots, s_n + 1)$

Logic

- Essentially Kripke semantics over the natural numbers

Logic

- Essentially Kripke semantics over the natural numbers
- Truth of a proposition depends on the step-index n
- P holds at n if it is true for n steps

Logic

- Essentially Kripke semantics over the natural numbers
- Truth of a proposition depends on the step-index n
- P holds at n if it is true for n steps
- If P is true for n steps, then it is also true for less than n steps
- Hence: a truth value is a downward closed subset of step indices
  Equivalently (classically), a conatural number

Propositions in $\mathcal{S}$

- Define the object Ω as

$$\{0,1\} \xleftarrow{r_0^\Omega} \{0,1,2\} \xleftarrow{r_1^\Omega} \{0,1,2,3\} \xleftarrow{r_2^\Omega} \cdots$$

  where $r_n^\Omega(m) = \min(m, n+1)$.
- $\triangleright : \Omega \to \Omega$ is given by $\triangleright_n(m) = \min(m+1, n+1)$

Propositions in $\mathcal{S}$

- Define the object Ω as

$$\{0,1\} \xleftarrow{r_0^\Omega} \{0,1,2\} \xleftarrow{r_1^\Omega} \{0,1,2,3\} \xleftarrow{r_2^\Omega} \cdots$$

  where $r_n^\Omega(m) = \min(m, n+1)$.
- $\triangleright : \Omega \to \Omega$ is given by $\triangleright_n(m) = \min(m+1, n+1)$

Propositions in $\mathcal{S}$

- Define the object Ω as

$$\{0,1\} \xleftarrow{r_0^\Omega} \{0,1,2\} \xleftarrow{r_1^\Omega} \{0,1,2,3\} \xleftarrow{r_2^\Omega} \cdots$$

  where $r_n^\Omega(m) = \min(m, n+1)$.
- $\triangleright : \Omega \to \Omega$ is given by $\triangleright_n(m) = \min(m+1, n+1)$
- Forcing relation: for $P : X \to \Omega$, $n \in \mathbb{N}$, and $x \in X_n$, we define

$$n, x \Vdash P \iff n \in P_n(x)$$

Propositions in $\mathcal{S}$

- Define the object Ω as

$$\{0,1\} \xleftarrow{r_0^\Omega} \{0,1,2\} \xleftarrow{r_1^\Omega} \{0,1,2,3\} \xleftarrow{r_2^\Omega} \cdots$$

  where $r_n^\Omega(m) = \min(m, n+1)$.
- $\triangleright : \Omega \to \Omega$ is given by $\triangleright_n(m) = \min(m+1, n+1)$
- Forcing relation: for $P : X \to \Omega$, $n \in \mathbb{N}$, and $x \in X_n$, we define

$$n, x \Vdash P \iff n \in P_n(x)$$

- Crucially: $n+1, x \Vdash \triangleright P \iff n, x|_n \Vdash P$

$\triangleright$ and quantifiers

- We have

$$\exists x : A.\ \triangleright P \vdash \triangleright(\exists x : A.\, P) \qquad\qquad \triangleright(\forall x : A.\, P) \vdash \forall x : A.\ \triangleright P$$

$\triangleright$ and quantifiers

- We have

$$\exists x : A.\ \triangleright P \vdash \triangleright(\exists x : A.\, P) \qquad\qquad \triangleright(\forall x : A.\, P) \vdash \forall x : A.\ \triangleright P$$

- However, the other directions are not valid:

$$n + 1 \Vdash \triangleright(\exists x : A.\, P) \iff \exists a \in A_n.\, n, a \Vdash P$$
$$n + 1 \Vdash \exists x : A.\ \triangleright P \iff \exists a \in A_{n+1}.\, n, a|_n \Vdash P$$

$\triangleright$ and quantifiers

- We have

$$\exists x : A.\ \triangleright P \vdash \triangleright(\exists x : A.\ P) \qquad \triangleright(\forall x : A.\ P) \vdash \forall x : A.\ \triangleright P$$

- However, the other directions are not valid:

$$n+1 \Vdash \triangleright(\exists x : A.\ P) \iff \exists a \in A_n.\ n, a \Vdash P$$
$$n+1 \Vdash \exists x : A.\ \triangleright P \iff \exists a \in A_{n+1}.\ n, a|_n \Vdash P$$

- There does not seem to be a general rule for commuting $\triangleright$ with a quantifier

Current solution

- Require that A be **total and inhabited**
  - Semantic condition: A_0 is inhabited and all r_n^A are surjective

Current solution

- Require that A be **total and inhabited**
  - Semantic condition: A_0 is inhabited and all r_n^A are surjective
- A is total and inhabited $\iff$ `next` is internally surjective:

$$\mathtt{TI}\,(A) := \forall y : \blacktriangleright A.\, \exists x : A.\, \mathtt{next}\, x = y$$

Current solution

- Require that A be **total and inhabited**
  - Semantic condition: A_0 is inhabited and all r_n^A are surjective
- A is total and inhabited $\iff$ `next` is internally surjective:

$$\mathtt{TI}\,(A) := \forall y : \blacktriangleright A.\, \exists x : A.\, \mathtt{next}\, x = y$$

- The following rule is valid in $\mathcal{S}$ [BMSS12]:

$$\frac{\vdash \mathtt{TI}\,(A) \qquad \Gamma, x : A \vdash P : \mathtt{Prop}}{\Gamma \mid \triangleright(\exists x : A.\, P) \vdash \exists x : A.\, \triangleright P}$$

Current solution

- Require that A be **total and inhabited**
  - Semantic condition: A_0 is inhabited and all r_n^A are surjective
- A is total and inhabited $\iff$ `next` is internally surjective:

$$\mathtt{TI}\,(A) := \forall y : \blacktriangleright A.\, \exists x : A.\, \mathtt{next}\, x = y$$

- The following rule is valid in $\mathcal{S}$ [BMSS12]:

$$\frac{\vdash \mathtt{TI}\,(A) \qquad \Gamma, x : A \vdash P : \mathtt{Prop}}{\Gamma \mid \triangleright(\exists x : A.\, P) \vdash \exists x : A.\ \triangleright P}$$

- Issue: only works for total and inhabited types

Current solution

- Require that A be **total and inhabited**
  - Semantic condition: A_0 is inhabited and all r_n^A are surjective
- A is total and inhabited $\iff$ $\texttt{next}$ is internally surjective:

$$\texttt{TI}\,(A) := \forall y : \blacktriangleright A.\, \exists x : A.\, \texttt{next}\, x = y$$

- The following rule is valid in $\mathcal{S}$ [BMSS12]:

$$\frac{\vdash \texttt{TI}\,(A) \qquad \Gamma, x : A \vdash P : \texttt{Prop}}{\Gamma \mid \triangleright(\exists x : A.\, P) \vdash \exists x : A.\, \triangleright P}$$

- Issue: only works for total and inhabited types
- Remark: following rule also valid in $\mathcal{S}$:

$$\frac{\Gamma, x : A \vdash P : \texttt{Prop}}{\Gamma \mid \texttt{TI}\,(A) \wedge \triangleright(\exists x : A.\, P) \vdash \exists x : A.\, \triangleright P}$$

lift

- We can decompose $\triangleright$ as $\triangleright = \texttt{lift} \circ \texttt{next}$ [BMSS12, CBGB16], where

$$\begin{aligned} \texttt{lift} \colon \blacktriangleright \Omega &\to \Omega \\ \texttt{lift}_0(*) &= 1 \\ \texttt{lift}_{n+1}(m) &= m + 1 \end{aligned}$$

lift

- We can decompose $\triangleright$ as $\triangleright = \texttt{lift} \circ \texttt{next}$ [BMSS12, CBGB16], where

$$\begin{aligned} \texttt{lift} \colon \blacktriangleright \Omega &\to \Omega \\ \texttt{lift}_0(*) &= 1 \\ \texttt{lift}_{n+1}(m) &= m + 1 \end{aligned}$$

- Hence, we could investigate the properties of lift

Novel commuting rules

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{ex} \circledast Q) \dashv\vdash \exists y : \blacktriangleright A.\, \texttt{lift}\,(Q \circledast y)}$$

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{all} \circledast Q) \dashv\vdash \forall y : \blacktriangleright A.\, \texttt{lift}\,(Q \circledast y)}$$

where

$$\texttt{ex} = \lambda P : A \to \texttt{Prop}.\, \exists x : A.\, P\, x$$
$$\texttt{all} = \lambda P : A \to \texttt{Prop}.\, \forall x : A.\, P\, x$$

Novel commuting rules

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{ex} \circledast Q) \dashv\vdash \exists y : \blacktriangleright A.\, \texttt{lift}\,(Q \circledast y)}$$

$$\frac{\Gamma \vdash Q : \blacktriangleright(A \to \texttt{Prop})}{\Gamma \mid \texttt{lift}\,(\texttt{next}\,\texttt{all} \circledast Q) \dashv\vdash \forall y : \blacktriangleright A.\, \texttt{lift}\,(Q \circledast y)}$$

where

$$\texttt{ex} = \lambda P : A \to \texttt{Prop}.\, \exists x : A.\, P\, x$$
$$\texttt{all} = \lambda P : A \to \texttt{Prop}.\, \forall x : A.\, P\, x$$

- Sound in $\mathcal{S}$
- Imply previous rules

Conclusion

The operation `lift` seems to be more fundamental and better behaved than $\triangleright$

Conclusion

The operation `lift` seems to be more fundamental and better behaved than $\triangleright$

Future work:

- Find appropriate rules for `lift`

Conclusion

The operation `lift` seems to be more fundamental and better behaved than $\triangleright$

Future work:

- Find appropriate rules for `lift`
- Investigate the applicability of the rules, e.g. by formalizing a model of Iris [JKJ+18] in this logic

Conclusion

The operation `lift` seems to be more fundamental and better behaved than $\triangleright$

Future work:

- Find appropriate rules for `lift`
- Investigate the applicability of the rules, e.g. by formalizing a model of Iris [JKJ+18] in this logic
- Check if the new rules also hold in other models of step-indexing (e.g. Transfinite Iris [SGG+21])

Conclusion

The operation `lift` seems to be more fundamental and better behaved than $\triangleright$

Future work:

- Find appropriate rules for `lift`
- Investigate the applicability of the rules, e.g. by formalizing a model of Iris [JKJ+18] in this logic
- Check if the new rules also hold in other models of step-indexing (e.g. Transfinite Iris [SGG+21])
- Connect `lift` to $\widehat{\triangleright} : \blacktriangleright U \to U$ in guarded dependent type theory [BGC+16]

Conclusion

The operation `lift` seems to be more fundamental and better behaved than $\triangleright$

Future work:

- Find appropriate rules for `lift`
- Investigate the applicability of the rules, e.g. by formalizing a model of Iris [JKJ+18] in this logic
- Check if the new rules also hold in other models of step-indexing (e.g. Transfinite Iris [SGG+21])
- Connect `lift` to $\widehat{\triangleright} : \blacktriangleright U \to U$ in guarded dependent type theory [BGC+16]
- Investigate analogues/generalisations in modal type theory [GKNB21]

References I

Amal J. Ahmed.
Semantics of types for mutable state.
PhD thesis, Princeton University, 2004.

Andrew W. Appel and David A. McAllester.
An indexed model of recursive types for foundational proof-carrying code.
ACM Trans. Program. Lang. Syst., 23(5):657–683, 2001.

Andrew W. Appel, Paul-André Melliès, Christopher D. Richards, and Jérôme Vouillon.
A very modal model of a modern, major, general type system.
In *POPL*, pages 109–122, 2007.

Ales Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal.
Guarded dependent type theory with coinductive types.
In *FoSSaCS*, volume 9634 of *LNCS*, pages 20–35, 2016.

References II

Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring.
First steps in synthetic guarded domain theory: step-indexing in the topos of trees.
Log. Methods Comput. Sci., 8(4), 2012.

Ranald Clouston, Ales Bizjak, Hans Bugge Grathwohl, and Lars Birkedal.
The guarded lambda-calculus: Programming and reasoning with guarded recursion for coinductive types.
Log. Methods Comput. Sci., 12(3), 2016.

Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal.
Multimodal dependent type theory.
Log. Methods Comput. Sci., 17(3), 2021.

References III

Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Ales Bizjak, Lars Birkedal, and Derek Dreyer.
Iris from the ground up: A modular foundation for higher-order concurrent separation logic.
J. Funct. Program., 28:e20, 2018.

Hiroshi Nakano.
A modality for recursion.
In *LICS*, pages 255–266, 2000.

Simon Spies, Lennard Gäher, Daniel Gratzer, Joseph Tassarotti, Robbert Krebbers, Derek Dreyer, and Lars Birkedal.
Transfinite Iris: Resolving an existential dilemma of step-indexed separation logic.
In *PLDI*, pages 80–95, 2021.