# Guarded Computational Type Theory

Jon Sterling
joint work with Robert Harper

Carnegie Mellon University

# what is $\mathrm{CTT}_{⏲}$?

a ***dependently typed*** program logic with guarded recursion, clocks, and coinductive types

# what is CTT$_{🕒}$?

a ***dependently typed*** program logic with guarded recursion, clocks, and coinductive types

- ⋆ operational account of guarded recursion

# what is CTT$_{\circlearrowleft}$?

a ***dependently typed*** program logic with guarded recursion, clocks, and coinductive types

- ⋆ operational account of guarded recursion
- ⋆ immediate canonicity result

# what is $CTT_{\circledcirc}$?

a ***dependently typed*** program logic with guarded recursion, clocks, and coinductive types

- ⋆ operational account of guarded recursion
- ⋆ immediate canonicity result
- ⋆ simple programming language (just $\lambda$-calculus)

# what is $\mathrm{CTT}_{🕒}$?

a ***dependently typed*** program logic with guarded recursion, clocks, and coinductive types

- ⋆ operational account of guarded recursion
- ⋆ immediate canonicity result
- ⋆ simple programming language (just $\lambda$-calculus)
- ⋆ defined by “operational sheaf model”

# what is CTT$_{🕒}$?

a ***dependently typed*** program logic with guarded recursion, clocks, and coinductive types

- ⋆ operational account of guarded recursion
- ⋆ immediate canonicity result
- ⋆ simple programming language (just λ-calculus)
- ⋆ defined by “operational sheaf model”

**COMPUTATION IS THE NEVER-SETTING SUN!**

# what is guarded recursion?

Introduced by Nakano, extended to dependent types in ***Guarded Dependent Type Theory*** (Bizjak et al).

# what is guarded recursion?

Introduced by Nakano, extended to dependent types in ***Guarded Dependent Type Theory*** (Bizjak et al).

$$\frac{A \;\; \text{type}}{\triangleright A \;\; \text{type}} \qquad \frac{M \in A}{\mathsf{next}(M) \in \triangleright A} \qquad \frac{x : \triangleright A \vdash M \in A}{\mathsf{fix}(x.M) \in A}$$

$$\frac{M \in \triangleright(A \to B) \qquad N \in \triangleright A}{M \circledast N \in \triangleright B}$$

# what is guarded recursion?

Introduced by Nakano, extended to dependent types in ***Guarded Dependent Type Theory*** (Bizjak et al).

$$\frac{A \ \text{type}}{\triangleright A \ \text{type}} \qquad \frac{M \in A}{\mathsf{next}(M) \in \triangleright A} \qquad \frac{x : \triangleright A \vdash M \in A}{\mathsf{fix}(x.M) \in A}$$

$$\frac{M \in \triangleright(A \to B) \quad N \in \triangleright A}{M \circledast N \in \triangleright B}$$

Solve “guarded recursive equations”:

$$\mathsf{Stream} \cong \mathbb{N} \times \triangleright\mathsf{Stream}$$

# semantics in the topos of trees

internal (extensional) type theory of $\mathcal{S} \triangleq \widehat{\omega}$.

# semantics in the topos of trees

internal (extensional) type theory of $\mathcal{S} \triangleq \widehat{\omega}$.

$$\triangleright : \mathcal{S} \to \mathcal{S}$$

$$\triangleright(X)(0) \triangleq \{*\}$$

$$\triangleright(X)(n+1) \triangleq X(n)$$

# semantics in the topos of trees

internal (extensional) type theory of $\mathcal{S} \triangleq \widehat{\omega}$.

$$\triangleright : \mathcal{S} \to \mathcal{S}$$

$$\triangleright(X)(0) \triangleq \{*\}$$

$$\triangleright(X)(n+1) \triangleq X(n)$$

(Straightforward to interpret $\mathsf{fix}(x.M)$.)

# causal (co)programming

the “later modality” $\triangleright$ enables ***causal*** programs on streams

$$\mathsf{incr} : \mathsf{Stream} \to \mathsf{Stream}$$
$$\mathsf{incr} \triangleq \mathrm{fix}(F.\ \lambda\langle n, \alpha\rangle.\ \langle n+1, F \circledast \alpha\rangle)$$

# causal (co)programming

the “later modality” $\triangleright$ enables ***causal*** programs on streams

$$\mathsf{incr} : \mathsf{Stream} \to \mathsf{Stream}$$
$$\mathsf{incr} \triangleq \mathsf{fix}(F.\ \lambda\langle n, \alpha\rangle.\ \langle n+1, F \circledast \alpha\rangle)$$

but not all ***productive*** operations are causal: delete every other item, etc.

# causal (co)programming

the "later modality" $\triangleright$ enables ***causal*** programs on streams

$$\mathsf{incr} : \mathsf{Stream} \to \mathsf{Stream}$$
$$\mathsf{incr} \triangleq \mathsf{fix}(F.\ \lambda\langle n, \alpha\rangle.\ \langle n+1, F \circledast \alpha\rangle)$$

but not all ***productive*** operations are causal: delete every other item, etc.

guarded recursion not enough for (co)programming!

# local clocks for coinduction

idea: parameterize $\triangleright$ by a ***clock*** $\kappa$ and add quantifier $\forall\kappa$.
due to McBride, Atkey

$$\frac{\Gamma \vdash_{\Delta,\kappa} M \in A[\kappa]}{\Gamma \vdash_{\Delta} \Lambda\kappa.M \in \forall\kappa.A[\kappa]} \qquad \frac{\Gamma \vdash_{\Delta} M \in \forall\kappa.A[\kappa] \quad \kappa \mathrel{\#} \Delta}{\Gamma \vdash_{\Delta} M[\kappa'] \in A[\kappa']}$$

# local clocks for coinduction

**idea**: parameterize $\triangleright$ by a ***clock*** $\kappa$ and add quantifier $\forall\kappa$.
due to McBride, Atkey

$$\frac{\Gamma \vdash_{\Delta,\kappa} M \in A[\kappa]}{\Gamma \vdash_{\Delta} \Lambda\kappa.M \in \forall\kappa.A[\kappa]} \qquad \frac{\Gamma \vdash_{\Delta} M \in \forall\kappa.A[\kappa]}{\Gamma \vdash_{\Delta} M[\kappa'] \in A[\kappa']}$$

# local clocks for coinduction

idea: parameterize $\triangleright$ by a ***clock*** $\kappa$ and add quantifier $\forall\kappa$.
due to McBride, Atkey

$$\frac{\Gamma \vdash_{\Delta,\kappa} M \in A[\kappa]}{\Gamma \vdash_{\Delta} \Lambda\kappa.M \in \forall\kappa.A[\kappa]} \qquad \frac{\Gamma \vdash_{\Delta} M \in \forall\kappa.A[\kappa]}{\Gamma \vdash_{\Delta} M[\kappa'] \in A[\kappa']}$$

$$\frac{\Gamma \vdash_{\Delta} A \ \text{type} \quad \kappa \in \Delta}{\Gamma \vdash_{\Delta} \triangleright_{\kappa} A \ \text{type}} \qquad \frac{\Gamma \vdash_{\Delta} M \in A}{\Gamma \vdash_{\Delta} \mathsf{next}_{\kappa}(M) \in \triangleright_{\kappa} A}$$

$$\frac{\Gamma, x : \triangleright_{\kappa} A \vdash_{\Delta} M \in A}{\Gamma \vdash_{\Delta} \mathsf{fix}_{\kappa}(x.M) \in A} \qquad \frac{M \in \triangleright(A \to B) \quad N \in \triangleright A}{M \circledast N \in \triangleright B}$$

# local clocks for coinduction

require the following isomorphisms:

$$\forall\kappa.\, \triangleright_{\kappa} A \cong \forall\kappa.A$$
$$\forall\kappa.A \cong A \quad (\kappa \# A)$$
$$\forall\kappa.A \times B \cong (\forall\kappa.A) \times (\forall\kappa.B)$$

etc.

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$

$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$
$$\mathsf{Stream} \triangleq \forall\kappa. \mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$\mathsf{Stream}$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$
$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$\forall\kappa.\mathsf{Stream}_\kappa$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$
$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$\forall\kappa.\mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$

$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$(\forall\kappa.\mathbb{N}) \times (\forall\kappa. \triangleright_\kappa \mathsf{Stream}_\kappa)$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$
$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$\mathbb{N} \times (\forall\kappa.\, \triangleright_\kappa \mathsf{Stream}_\kappa)$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$
$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$\mathbb{N} \times (\forall\kappa.\mathsf{Stream}_\kappa)$$

# $\triangleright_\kappa$ + $\forall\kappa$ = coinduction!

Define streams in two parts (decompose productivity into causality and free use):

$$\mathsf{Stream}_\kappa \cong \mathbb{N} \times \triangleright_\kappa \mathsf{Stream}_\kappa$$
$$\mathsf{Stream} \triangleq \forall\kappa.\mathsf{Stream}_\kappa$$

We have $\mathsf{Stream} \cong \mathbb{N} \times \mathsf{Stream}$:

$$\mathbb{N} \times \mathsf{Stream}$$

# intensional version?

Both computation rule for $\mathsf{fix}_\kappa(x.M)$ and extensionality rule for $\mathsf{next}_\kappa$ incompatible with standard intensional type theory.

Birkedal et al resolve both problems in Guarded Cubical Type Theory. (Conjectured decidable typing result.)

**canonicity currently unknown; no operational semantics**
(but stay tuned)

# resurrecting extensionality

In $\mathbf{CTT}_{\circlearrowleft}$ we contribute a version of Guarded Dependent Type Theory with simple operational semantics, immediate canonicity result, simpler term language, and new type equations.

# resurrecting extensionality

In **CTT**$_{\circledcirc}$ we contribute a version of Guarded Dependent Type Theory with simple operational semantics, immediate canonicity result, simpler term language, and new type equations.

Hope to combine with Computational Higher Dimensional Type Theory (Angiuli, Harper, Wilson). Guarded recursion will be the first of many extensions.

# preview of CTT$_{\circlearrowright}$

$$\frac{\Gamma \gg_{\Delta} M \in A}{\Gamma \gg_{\Delta} M \in \triangleright_{\kappa} A} \qquad \frac{\Gamma \gg_{\Delta} M \in \triangleright_{\kappa}(\Pi x : A.B[x]) \quad \Gamma \gg_{\Delta} M \in \triangleright_{\kappa} A}{\Gamma \gg_{\Delta} M(N) \in \triangleright_{\kappa} B[N]}$$

(look ma, no delayed substitutions!)

$$\frac{\Gamma, x : \triangleright_{\kappa} A \gg_{\Delta} M \in A}{\Gamma \gg_{\Delta} \mathsf{fix}(x.M) \in A}$$

# preview of CTT$_{\text{🕒}}$

$$\frac{\Gamma \gg_{\Delta} M \in A}{\Gamma \gg_{\Delta} M \in \triangleright_{\kappa} A} \qquad \frac{\Gamma \gg_{\Delta} M \in \triangleright_{\kappa}(\Pi x : A.B[x]) \quad \Gamma \gg_{\Delta} M \in \triangleright_{\kappa} A}{\Gamma \gg_{\Delta} M(N) \in \triangleright_{\kappa} B[N]}$$

$$\frac{\Gamma, x : \triangleright_{\kappa} A \gg_{\Delta} M \in A}{\Gamma \gg_{\Delta} \mathsf{fix}(x.M) \in A}$$

# preview of CTT$_{⏲}$

$$\frac{\Gamma \gg_\Delta M \in A}{\Gamma \gg_\Delta M \in \triangleright_\kappa A} \qquad \frac{\Gamma \gg_\Delta M \in \triangleright_\kappa(\Pi x : A.B[x]) \quad \Gamma \gg_\Delta M \in \triangleright_\kappa A}{\Gamma \gg_\Delta M(N) \in \triangleright_\kappa B[N]}$$

$$\frac{\Gamma, x : \triangleright_\kappa A \gg_\Delta M \in A}{\Gamma \gg_\Delta \mathsf{fix}(x.M) \in A}$$

$$\frac{\Gamma \gg_{\Delta,\kappa} M \in A}{\Gamma \gg_\Delta M \in \cap\kappa.A} \qquad \frac{\Gamma \gg_\Delta M \in \cap\kappa.A}{\Gamma \gg_\Delta M \in A[\kappa]}$$

# preview of CTT$_{🕒}$

$$\frac{\Gamma \gg_\Delta M \in A}{\Gamma \gg_\Delta M \in \triangleright_\kappa A} \qquad \frac{\Gamma \gg_\Delta M \in \triangleright_\kappa(\Pi x : A.B[x]) \quad \Gamma \gg_\Delta M \in \triangleright_\kappa A}{\Gamma \gg_\Delta M(N) \in \triangleright_\kappa B[N]}$$

$$\frac{\Gamma, x : \triangleright_\kappa A \gg_\Delta M \in A}{\Gamma \gg_\Delta \mathsf{fix}(x.M) \in A}$$

$$\frac{\Gamma \gg_{\Delta,\kappa} M \in A}{\Gamma \gg_\Delta M \in \cap\kappa.A} \qquad \frac{\Gamma \gg_\Delta M \in \cap\kappa.A}{\Gamma \gg_\Delta M \in A[\kappa]}$$

use extensionality as a weapon!

$$\cap\kappa.\triangleright_\kappa A \doteq \cap\kappa.A \;\; \mathsf{type}$$

$$\cap\kappa.A \doteq A \;\; \mathsf{type} \quad (\kappa \# A)$$

$$\cap\kappa.A \times B \doteq (\cap\kappa.A) \times (\cap\kappa.B) \;\; \mathsf{type}$$

# topos semantics

We develop the computational PER semantics (meaning explanation) for **CTT**$_{\circleddash}$ internally to a suitable ***sheaf topos*** which provides all the machinery that we need ($\triangleright_\kappa$, $\forall\kappa$).

# topos semantics

We develop the computational PER semantics (meaning explanation) for **CTT**$_{\circledcirc}$ internally to a suitable ***sheaf topos*** which provides all the machinery that we need ($\triangleright_\kappa$, $\forall\kappa$).

Easier to do normal math than to wrestle with syntax! Do important proofs first, then implement PER semantics using the internal language of the topos.

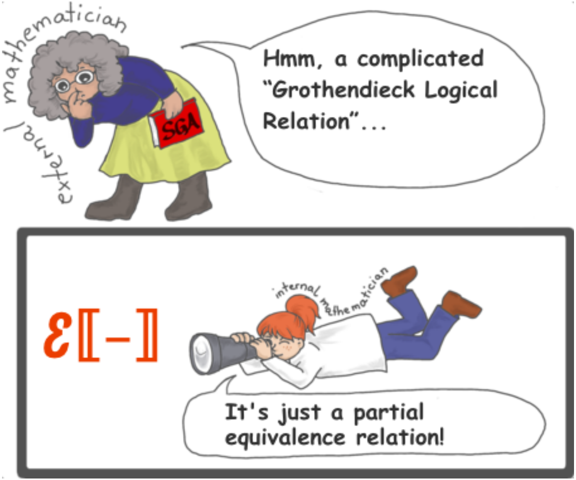
# topos semantics

We develop the computational PER semantics (meaning explanation) for **CTT**$_{\circlearrowright}$ internally to a suitable ***sheaf topos*** which provides all the machinery that we need ($\triangleright_\kappa$,$\forall\kappa$).

Easier to do normal math than to wrestle with syntax! Do important proofs first, then implement PER semantics using the internal language of the topos.

Please use these slogans:

- ⋆ **CRITICIZE THE OLD WORLD AND BUILD A NEW ONE WITH INTERNAL THOUGHT AS A WEAPON!**
- ⋆ **UPHOLD BETH-KRIPKE-JOYAL THOUGHT!**




(credit: Ingo Blechschmidt)

(time to use the blackboard)