Guarded recursion in the topos of trees

Bálint Kocsis

Radboud University

March 7, 2025

Step-indexing

- Problem: non-well-founded definitions, e.g.

$$X = X \Rightarrow A$$

Step-indexing

- Problem: non-well-founded definitions, e.g.

$$X = X \Rightarrow A$$

- Solution: stratify recursion using sequences of successive approximations

$$\begin{aligned} X_{-1} &= V \\ X_{n+1} &= X_n \cap (X_n \Rightarrow A) \\ X &= \bigcap X_n \end{aligned}$$

Step-indexing

- Problem: non-well-founded definitions, e.g.

$$X = X \Rightarrow A$$

- Solution: stratify recursion using sequences of successive approximations

$$\begin{aligned} X_{-1} &= V \\ X_{n+1} &= X_n \cap (X_n \Rightarrow A) \\ X &= \bigcap X_n \end{aligned}$$

- Syntactically: introduce modality $\blacktriangleright$ to talk about the next step

$$T = \mu X. \blacktriangleright X \to A$$

Applications

- Solve recursive domain equations
- Model general recursive types
- Model general references
- Define recursive functions, including negative self-references
- Concrete projects
    - Typed intermediate/assembly languages: FPCC
    - Program logics: VST, Iris
    - Guarded type theory

Focus of my work

- Logic of step-indexing/guarded recursion
- Particular model: topos of trees
- Formalization in Coq

Guarded type theory

- New type former $\blacktriangleright$
  - Allows us to talk about data we will only have access to in the next computation step
  - Guards self-references in type/term definitions:

$$T = \mu X. \blacktriangleright X \to A$$

Guarded type theory

- New type former $\blacktriangleright$
  - Allows us to talk about data we will only have access to in the next computation step
  - Guards self-references in type/term definitions:

$$T = \mu X. \blacktriangleright X \to A$$

- Applicative structure
  - $\mathtt{next} : A \to \blacktriangleright A$
  - $- \circledast - : \blacktriangleright (A \to B) \to \blacktriangleright A \to \blacktriangleright B$

Guarded type theory

- New type former $\blacktriangleright$
  - Allows us to talk about data we will only have access to in the next computation step
  - Guards self-references in type/term definitions:

$$T = \mu X.\blacktriangleright X \to A$$

- Applicative structure
  - $\texttt{next} : A \to \blacktriangleright A$
  - $- \circledast - : \blacktriangleright(A \to B) \to \blacktriangleright A \to \blacktriangleright B$
- Guarded fixed point combinator: $\texttt{fix} : (\blacktriangleright A \to A) \to A$
  - Self-reference delayed in time by $\texttt{next}$:

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

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

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:

$$
\begin{aligned}
- :: - &: \mathbb{N} \to \blacktriangleright \texttt{Str} \to \texttt{Str} \\
\texttt{hd} &: \texttt{Str} \to \mathbb{N} \\
\texttt{tl} &: \texttt{Str} \to \blacktriangleright \texttt{Str}
\end{aligned}
$$

- Recursive operations:

$$
\begin{aligned}
\texttt{zeros} &= \mu s.0 :: s : \texttt{Str} \\
\texttt{add} &= \lambda n.\mu r.\lambda s.(\texttt{hd}\, s + n) :: (r \circledast \texttt{tl}\, s) : \mathbb{N} \to \texttt{Str} \to \texttt{Str}
\end{aligned}
$$

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.

- $\blacktriangleright$ type former $\Rightarrow$ $\triangleright$ modality
  - $\triangleright P$ holds 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
  - $\triangleright P$ holds if P holds at the next step
- Applicative structure $\Rightarrow$ modal axioms:

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

Step-indexed logic

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

- $\blacktriangleright$ type former $\Rightarrow$ $\triangleright$ modality
  - $\triangleright P$ holds if P holds at the next step
- Applicative structure $\Rightarrow$ modal axioms:

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

- 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

Some important rules

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

$$\triangleright(P * Q) \dashv\vdash \triangleright P * \triangleright Q \quad (* \in \{\wedge, \vee, \supset\})$$

$$\triangleright(t =_A u) \dashv\vdash \mathtt{next}\, t =_{\blacktriangleright A} \mathtt{next}\, u$$

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
- Two main formalisms:
  - Ordered families of equivalences (used by Iris): a set equipped with more and more refined equivalence relations
  - Topos of trees: sequence of sets with restriction maps

Topos of trees

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

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

- Morphisms $f : X \to Y$:

$$\begin{tikzcd}
X_0 \arrow[d, "f_0"'] & X_1 \arrow[l, "r_0^X"'] \arrow[d, "f_1"'] & X_2 \arrow[l, "r_1^X"'] \arrow[d, "f_2"'] & \cdots \arrow[l, "r_2^X"'] \\
Y_0 & Y_1 \arrow[l, "r_0^Y"'] & Y_2 \arrow[l, "r_1^Y"'] & \cdots \arrow[l, "r_2^Y"']
\end{tikzcd}$$

Guarded recursion

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

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

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

$$\begin{tikzcd}
X_0 \arrow[d, "!"'] & X_1 \arrow[l, "r_0^X"'] \arrow[d, "r_0^X"'] & X_2 \arrow[l, "r_1^X"'] \arrow[d, "r_1^X"'] & \cdots \arrow[l, "r_2^X"'] \\
\{*\} & X_0 \arrow[l, "!"] & X_1 \arrow[l, "r_0^X"] & \cdots \arrow[l, "r_1^X"]
\end{tikzcd}$$

Examples

- Streams:

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

- $\mathtt{hd}_n(s_0, \ldots, s_n) = s_0$
- $\mathtt{inc}_n(s_0, \ldots, s_n) = (s_0 + 1, \ldots, s_n + 1)$

Guarded fixed points

Proposition

Let $f : \blacktriangleright X \to X$ be a morphism of $\mathcal{S}$. Then there exists a unique $x : \mathbf{1} \to X$ such that $f \circ \mathtt{next}_X \circ x = x$.

Proof.

We have $f_0 : \{*\} \to X_0$ and $f_{n+1} : X_n \to X_{n+1}$. Define $x : \mathbf{1} \to X$ by recursion: $x_0 = f_0$ and $x_{n+1} = f_{n+1} \circ x_n$. □

Logic

- Essentially Kripke semantics over the natural numbers
- Intuition: the truth of a proposition depends on the step-index n, i.e. the amount of computation steps left
- 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

Kripke-Joyal semantics

Forcing relation: $n \Vdash P$ iff P holds at step n

Proposition

The forcing relation satisfies the following clauses:

$$n \Vdash P \supset Q \textit{ iff } \forall m \le n : m \Vdash P \Rightarrow m \Vdash Q$$
$$n \Vdash \exists x : A.P \textit{ iff } \exists a \in [\![A]\!]_n : n \Vdash P(a)$$
$$n \Vdash \forall x : A.P \textit{ iff } \forall m \le n, a \in [\![A]\!]_m : m \Vdash P(a)$$
$$n \Vdash \triangleright P \textit{ iff } n = 0 \vee n - 1 \Vdash P$$

$\triangleright$ 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, e.g.

$$n+1 \Vdash \triangleright(\exists x : A.P) \text{ iff } \exists a \in [\![A]\!]_n : n \Vdash P(a)$$
$$n+1 \Vdash \exists x : A. \triangleright P \text{ iff } \exists a \in [\![A]\!]_{n+1} : n \Vdash P(a|_n)$$

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

$\texttt{lift}$

- We can decompose $\triangleright = \texttt{lift} \circ \texttt{next}$ [1]
- Hence, we could investigate the properties of $\texttt{lift}$
- Novel rule:

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

where

$$Q : \blacktriangleright(A \to \texttt{Prop})$$

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

Coq formalization

- Need finite types for the definition of propositions
- Usual representation:

```
Inductive fin : nat → Type :=
  | FZ {n} : fin (S n)
  | FS {n} : fin n → fin (S n).
```

- Alternatively:

```
Definition fin (n : nat) := {m : nat | m < n}.
```

- The latter definition works much better in practice

Conclusion

- Exposition of the topos of trees
- Emphasis on `lift`, which seems to be more fundamental
- Coq formalization: case study in using proof-irrelevant propositions for the representation of finite types

Future work

- Find appropriate rules for `lift`
- Investigate step-indexed logic from the perspective of modal type theory
- Formalize a model of Iris in guarded type theory

References

R. Clouston, A. Bizjak, H. B. Grathwohl, and L. Birkedal.
The guarded lambda-calculus: Programming and reasoning with guarded recursion for coinductive types.
Logical Methods in Computer Science, Volume 12, Issue 3, Apr. 2017.