Canonicity for Indexed Inductive-Recursive Types

András Kovács

University of Gothenburg & Chalmers University of Technology

15th Jan 2026, POPL, Rennes

Inductive-Recursive Types

Small Agda example.

$$
\begin{aligned}
&\textbf{data}\ \mathsf{Code} : \mathsf{Set}\ \textbf{where} \\
&\quad \mathbb{N}' : \mathsf{Code} \\
&\quad \Pi' : (A : \mathsf{Code}) \to (\mathsf{El}\, A \to \mathsf{Code}) \to \mathsf{Code}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{El} : \mathsf{Code} \to \mathsf{Set} \\
&\mathsf{El}\, \mathbb{N}' \qquad = \mathbb{N} \\
&\mathsf{El}\, (\Pi'\, A\, B) = (a : \mathsf{El}\, A) \to \mathsf{El}\, (B\, a)
\end{aligned}
$$

Early and informal use by Per Martin-Löf [ML75, ML84].

Formal syntax & semantics developed by Dybjer & Setzer [Dyb00, DS99, DS03, DS06].

Inductive-Recursive Types

Let's parameterize Code and El with $\boldsymbol{u}$ **: Set** and **el :** $\boldsymbol{u} \longrightarrow$ **Set**, and add extra rules.

data Code : Set **where**

...

u' : Code

el' : $u \to$ Code

,

El : Code $\to$ Set

...

$\mathsf{El}\,\mathsf{u}' = u$

$\mathsf{El}\,(\mathsf{el}'\,a) = el\,a$

Inductive-Recursive Types

Let's parameterize Code and El with $\boldsymbol{u}$ **: Set** and **el :** $\boldsymbol{u} \to$ **Set**, and add extra rules.

data Code : Set **where**

...

u' : Code

el' : $u \to$ Code

'

El : Code $\to$ Set

...

$\mathsf{El}\,\mathsf{u}' = u$

$\mathsf{El}\,(\mathsf{el}'\,a) = el\,a$

Code and El yield a function F : FamSet $\to$ FamSet.

We get countable universes by iteration:

$$\begin{aligned} \mathsf{U} &: \mathbb{N} \to \mathsf{FamSet} \\ \mathsf{U}\,\mathsf{zero} &= (\bot, \mathsf{exfalso}) \\ \mathsf{U}\,(\mathsf{suc}\,n) &= \mathsf{F}\,(\mathsf{U}\,n) \end{aligned}$$

More generally: any well-founded universe hierarchy can be defined by induction-recursion inside Set.

Inductive-Recursive Types

IR is convenient for doing metatheory of universe features.

Recent works on various notions of first-class universe levels, all using IR in Agda formalizations:

- Consistency [Kov22]
- Canonicity [CW25]
- Here at POPL: normalization [DFK26]

Some other uses: partial functions [BC01], generic programming [BDJ03, Die17].

Implementation & Metatheory

IR has been supported in Agda for 15+ years. Also supported in Idris-es.

Intuitively, there is *nothing special* about the implementation.

Implementation & Metatheory

IR has been supported in Agda for 15+ years. Also supported in Idris-es.

Intuitively, there is *nothing special* about the implementation.

However, syntactic properties had been missing:

1. *Canonicity:* every closed term can be computed to a constructor.
2. *Normalization:* every open term can be computed to a normal form.

We show **canonicity** in the current work.

Canonicity: basic setup

We only talk about well-typed syntax, quotiented by conversion.

The syntax is given as a quotient inductive-inductive type with the following sorts:

$$
\begin{aligned}
&\mathsf{Con} : \mathsf{Set} \\
&\mathsf{Ty} \ : \mathsf{Con} \to \mathbb{N} \to \mathsf{Set} \\
&\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
&\mathsf{Tm} \ : (\Gamma : \mathsf{Con})\{i : \mathbb{N}\} \to \mathsf{Ty}\,\Gamma\,i \to \mathsf{Set}
\end{aligned}
$$

We have

- An explicit substitution calculus, given as a category with families.
- Russell-style universes, as $\mathsf{U} : (i : \mathbb{N}) \to \mathsf{Ty}\,\Gamma\,(i+1)$ such that $\mathsf{Tm}\,\Gamma\,(\mathsf{U}\,i) = \mathsf{Ty}\,\Gamma\,i$.
- Basic type formers: Π, Σ, $\top$, Bool, intensional identity.

Canonicity: basic setup

Artin gluing is an established proof-relevant version of **logical predicates** which works nicely for our syntax.[1]

We define a family of functions $-^\circ$ from the syntax by induction.

$$
\begin{aligned}
-^\circ &: (\Gamma : \mathsf{Con}) && \to \mathsf{Sub}\bullet\Gamma \to \mathsf{Set}_\omega \\
-^\circ &: (A : \mathsf{Ty}\,\Gamma\,i) && \to \{\gamma : \mathsf{Sub}\bullet\Gamma\}(\gamma^\circ : \Gamma^\circ\,\gamma) \to \mathsf{Tm}\bullet A[\gamma] \to \mathsf{Set}_i \\
-^\circ &: (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) && \to \{\gamma : \mathsf{Sub}\,\Gamma\}(\gamma^\circ : \Gamma^\circ\,\gamma) \to \Delta^\circ\,(\sigma \circ \gamma) \\
-^\circ &: (t : \mathsf{Tm}\,\Gamma\,A) && \to \{\gamma : \mathsf{Sub}\,\Gamma\}(\gamma^\circ : \Gamma^\circ\,\gamma) \to A^\circ\,\gamma^\circ\,t[\gamma]
\end{aligned}
$$

[1] See e.g. [Coq19, KHS19]

Canonicity for inductive types

The main task is to specify what it means for a closed term to be canonical.

Example: natural numbers.

In the syntax, assume $\mathbb{N} : \mathsf{Ty}\,\Gamma$ with zero and suc.

Closed canonical terms of type $\mathbb{N}$ are specified by an indexed inductive type in the metatheory.

$$
\begin{aligned}
&\textbf{data}\ \mathbb{N}^\circ : \mathsf{Tm}\,\bullet\,\mathbb{N} \to \mathsf{Set}\ \textbf{where}\\
&\quad \mathsf{zero}^\circ : \mathbb{N}^\circ\ \mathsf{zero}\\
&\quad \mathsf{suc}^\circ\ \ : (n : \mathsf{Tm}\,\bullet\,\mathbb{N}) \to \mathbb{N}^\circ\ n \to \mathbb{N}^\circ\ (\mathsf{suc}\ n)
\end{aligned}
$$

Canonicity for inductive types

Example: the canonicity predicate for the IR type Code is an indexed IR type in the metatheory.

Canonicity for inductive types

Example: the canonicity predicate for the IR type Code is an indexed IR type in the metatheory.

$$
\begin{aligned}
&\textbf{data}\ \mathsf{Code}^\circ : \mathsf{Tm}\bullet\mathsf{Code} \to \mathsf{Set}\\
&\quad \mathbb{N}'^\circ : \mathsf{Code}^\circ\,\mathbb{N}'\\
&\quad \Pi'^\circ : \{A : \mathsf{Tm}\bullet\mathsf{Code}\}(A^\circ : \mathsf{Code}^\circ\,A)\\
&\qquad \{B : \mathsf{Tm}\bullet(\mathsf{El}\,A \to \mathsf{Code})\}(B^\circ : \{a : \mathsf{Tm}\bullet(\mathsf{El}\,a)\} \to \mathsf{El}^\circ\,A^\circ\,a \to \mathsf{Code}^\circ\,(B\,a))\\
&\qquad \to \mathsf{Code}^\circ\,(\Pi'\,A\,B)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{El}^\circ : \{t : \mathsf{Tm}\bullet\mathsf{Code}\} \to \mathsf{Code}^\circ\,t \to (\mathsf{Tm}\bullet(\mathsf{El}\,t) \to \mathsf{Set})\\
&\mathsf{El}^\circ\,\mathbb{N}'^\circ \qquad\quad = \mathbb{N}^\circ\\
&\mathsf{El}^\circ\,(\Pi'^\circ\,A^\circ\,B^\circ) = \lambda\,f.\,\{a : \mathsf{Tm}\bullet(\mathsf{El}\,A)\}(a^\circ : \mathsf{El}^\circ\,A^\circ\,a) \to \mathsf{El}^\circ\,(B^\circ\,a^\circ)\,(f\,a)
\end{aligned}
$$

The type of **IR signatures** is an inductive type.

$$
\begin{aligned}
&\textbf{data}\ \mathsf{Sig}\ i\ \{j\}\ (O : \mathsf{Set}_j) : \mathsf{Set}_{(i+1 \sqcup j)}\ \textbf{where}\\
&\quad \iota\ : O \to \mathsf{Sig}\ i\ O\\
&\quad \sigma : (A : \mathsf{Set}_i) \to (A \to \mathsf{Sig}\ i\ O) \to \mathsf{Sig}\ i\ O\\
&\quad \delta\ : (A : \mathsf{Set}_i) \to ((A \to O) \to \mathsf{Sig}\ i\ O) \to \mathsf{Sig}\ i\ O
\end{aligned}
$$

The signature of the previous Code example:

$$
\begin{aligned}
&\mathsf{SigCode} : \mathsf{Sig}\ 0\ \mathsf{Set}\\
&\mathsf{SigCode} :\equiv \sigma\ \mathsf{Bool}\ \lambda\ t.\ \textbf{case}\ t\ \textbf{of}\\
&\quad \mathsf{true}\ \to \iota\ \mathbb{N}\\
&\quad \mathsf{false} \to \delta\ \top\ \lambda\ \mathit{ElA}.\ \delta\ (\mathit{ElA}\ \mathsf{tt})\ \lambda\ \mathit{ElB}.\ \iota\ ((x : \mathit{ElA}\ \mathsf{tt}) \to \mathit{ElB}\ x)
\end{aligned}
$$

For each signature, we postulate type formation, term formation, elimination and computation rules, for the described IR type.

How to prove canonicity?

Every closed term with IR type should be convertible to an IR constructor. E.g. a closed term t : Code should be either $\mathbb{N}'$ or Π'.

Type-theoretic Artin gluing [Coq19, KHS19]:

- A closed type is interpreted as a proof-relevant predicate over its closed terms. The predicate should imply canonicity.
- A closed term is interpreted as a predicate witness.
- Open types & terms are generalized over closed substitutions.

How to prove canonicity?

Notation: we write **Ty** for the set of closed types, **Tm *A*** for sets of closed terms, and **U : Ty** for object-theoretic universes (omitting levels).

Example: canonicity predicate for natural numbers.

$$\begin{aligned}&\textbf{data}\ \mathbb{N}^\circ : \mathsf{Tm}\,\mathbb{N} \to \mathsf{Set}\ \textbf{where}\\ &\quad \mathsf{zero}^\circ : \mathbb{N}^\circ\,\mathsf{zero}\\ &\quad \mathsf{suc}^\circ\ : (n : \mathsf{Tm}\,\mathbb{N}) \to \mathbb{N}^\circ\, n \to \mathbb{N}^\circ\,(\mathsf{suc}\, n)\end{aligned}$$

Generally: the canonicity predicate for an inductive type is a metatheoretic indexed inductive type.

How to prove canonicity?

Example: the canonicity predicate for Code is the following metatheoretic indexed IR type:

$$
\begin{aligned}
&\textbf{data}\ \mathsf{Code}^\circ : \mathsf{Tm}\,\mathsf{Code} \to \mathsf{Set}\\
&\quad \mathbb{N}'^\circ : \mathsf{Code}^\circ\,\mathbb{N}'\\
&\quad \Pi'^\circ : \{A : \mathsf{Tm}\,\mathsf{Code}\}(A^\circ : \mathsf{Code}^\circ\,A)\\
&\qquad \{B : \mathsf{Tm}\,(\mathsf{El}\,A \to \mathsf{Code})\}(B^\circ : \{a : \mathsf{Tm}\,(\mathsf{El}\,a)\} \to \mathsf{El}^\circ\,A^\circ\,a \to \mathsf{Code}^\circ\,(B\,a))\\
&\qquad \to \mathsf{Code}^\circ\,(\Pi'\,A\,B)
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{El}^\circ : \{t : \mathsf{Tm}\,\mathsf{Code}\} \to \mathsf{Code}^\circ\,t \to (\mathsf{Tm}\,(\mathsf{El}\,t) \to \mathsf{Set})\\
&\mathsf{El}^\circ\,\mathbb{N}'^\circ \qquad\quad = \mathbb{N}^\circ\\
&\mathsf{El}^\circ\,(\Pi'^\circ\,A^\circ\,B^\circ) = \lambda\,f.\{a : \mathsf{Tm}\,(\mathsf{El}\,A)\}(a^\circ : \mathsf{El}^\circ\,A^\circ\,a) \to \mathsf{El}^\circ\,(B^\circ\,a^\circ)\,(f\,a)
\end{aligned}
$$

If we have $t : \mathsf{Tm}\,\mathsf{Code}$, the gluing interpretation hands us an element of $\mathsf{Code}^\circ\,t$.

How to prove canonicity?

In the object theory, let's assume general IR types specified by an internal Sig type.

- Previous slide: the object theory supports Code, the canonicity proof involves Code°.

How to prove canonicity?

In the object theory, let's assume general IR types specified by an internal Sig type.

- Previous slide: the object theory supports Code, the canonicity proof involves Code°.
- Now: the object theory has all IR types, the canonicity proof involves all canonicity predicates.

How to prove canonicity?

In the object theory, let's assume general IR types specified by an internal Sig type.

- Previous slide: the object theory supports Code, the canonicity proof involves Code°.
- Now: the object theory has all IR types, the canonicity proof involves all canonicity predicates.

Interpreting IR types in the glued model:

- We write $\mathsf{IR}\,S : \mathsf{Tm}\,\mathsf{U}$ for the IR *type formation* rule, for $S : \mathsf{Tm}\,(\mathsf{Sig}\,O)$.

How to prove canonicity?

In the object theory, let's assume general IR types specified by an internal Sig type.

- Previous slide: the object theory supports Code, the canonicity proof involves Code°.
- Now: the object theory has all IR types, the canonicity proof involves all canonicity predicates.

Interpreting IR types in the glued model:

- We write $\mathsf{IR}\,S : \mathsf{Tm}\,\mathsf{U}$ for the IR *type formation* rule, for $S : \mathsf{Tm}\,(\mathsf{Sig}\,O)$.
- To interpret $\mathsf{IR}\,S$:
  - By induction hypothesis, we get $S^\circ : \mathsf{Sig}^\circ\,S$ witnessing the canonicity of the signature S itself.
  - We compute a metatheoretic indexed IR signature by induction on S°. This yields the canonicity predicate for $\mathsf{IR}\,S$.

How to prove canonicity?

In the object theory, let's assume general IR types specified by an internal Sig type.

- Previous slide: the object theory supports Code, the canonicity proof involves Code°.
- Now: the object theory has all IR types, the canonicity proof involves all canonicity predicates.

Interpreting IR types in the glued model:

- We write $\mathsf{IR}\,S : \mathsf{Tm}\,\mathsf{U}$ for the IR *type formation* rule, for $S : \mathsf{Tm}\,(\mathsf{Sig}\,O)$.
- To interpret $\mathsf{IR}\,S$:
  - By induction hypothesis, we get $S^\circ : \mathsf{Sig}^\circ\,S$ witnessing the canonicity of the signature S itself.
  - We compute a metatheoretic indexed IR signature by induction on S°. This yields the canonicity predicate for $\mathsf{IR}\,S$.
- We also need to interpret *term formation*, *elimination* and *computation* rules.
- For these, we have to show that universal properties of IR types are preserved through the indexed IR encoding.

How to prove canonicity?

The construction is a moderately technical "generic programming" exercise, where we have to do some tricky induction over signatures.

We use metatheoretic IR to show canonicity of object-theoretic IR. The metatheory has to have more universes; in our case the metatheory has extra levels ω and $\omega + 1$[2].

The canonicity interpretation of IR is formalized in Agda using a shallow embedding of Ty and Tm.

[2] But $\omega + 1$ is only used for convenience and could be omitted.

Indexed Induction-Recursion

What about canonicity for **indexed** IR types?

We only show canonicity for plain IR types, but also show that indexed IR types are constructible in MLTT+IR.

The construction supports strict computation rules for all IIR types that are definable in Agda. For *neutral* signatures we only get propositional computation.

We use the well-known construction that converts indices to parameters and propositional identities.[3]

Here, since we work in plain MLTT without function extensionality and UIP, we have to do a modest amount of HoTT reasoning.

[3] Popularized as "fording" by Conor McBride.

Closing notes

- Future work:
  - Normalization for IR types. For this, we first have to show that presheaf models have IR types (which is already quite technical!).
  - Generalize the canonicity proof to a model construction (i.e. extend general type-theoretic gluing with IR types). This seems to be incompatible with first-class signatures.
- Paper: under review. I'll publish a preprint in $\sim$2 weeks.

Closing notes

- Future work:
  - Normalization for IR types. For this, we first have to show that presheaf models have IR types (which is already quite technical!).
  - Generalize the canonicity proof to a model construction (i.e. extend general type-theoretic gluing with IR types). This seems to be incompatible with first-class signatures.
- Paper: under review. I'll publish a preprint in $\sim$2 weeks.

Thank you!

Ana Bove and Venanzio Capretta.
Nested general recursion and partiality in type theory.
In Richard J. Boulton and Paul B. Jackson, editors, *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001, Edinburgh, Scotland, UK, September 3-6, 2001, Proceedings*, volume 2152 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 2001.

Marcin Benke, Peter Dybjer, and Patrik Jansson.
Universes for generic programs and proofs in dependent type theory.
Nord. J. Comput., 10(4):265–289, 2003.

Thierry Coquand.
Canonicity and normalization for dependent type theory.
Theor. Comput. Sci., 777:184–191, 2019.

Jonathan Chan and Stephanie Weirich.
Bounded first-class universe levels in dependent type theory.
CoRR, abs/2502.20485, 2025.

Nils Anders Danielsson, Naïm Camille Favier, and Ondřej Kubánek.
Normalisation for first-class universe levels.
Proc. ACM Program. Lang., 10(POPL), January 2026.

Larry Diehl.
Fully Generic Programming over Closed Universes of Inductive-Recursive Types.

PhD thesis, Portland State University, 2017.

Peter Dybjer and Anton Setzer.
A finite axiomatization of inductive-recursive definitions.
In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings*, volume 1581 of *Lecture Notes in Computer Science*, pages 129–146. Springer, 1999.

Peter Dybjer and Anton Setzer.
Induction-recursion and initial algebras.
Ann. Pure Appl. Log., 124(1-3):1–47, 2003.

Peter Dybjer and Anton Setzer.
Indexed induction-recursion.
J. Log. Algebraic Methods Program., 66(1):1–49, 2006.

Peter Dybjer.
A general formulation of simultaneous inductive-recursive definitions in type theory.
Journal of Symbolic Logic, 65:525–549, 2000.

Ambrus Kaposi, Simon Huber, and Christian Sattler.
Gluing for type theory.

In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*, volume 131 of *LIPIcs*, pages 25:1–25:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

András Kovács.
Generalized universe hierarchies and first-class universe levels.
In Florin Manea and Alex Simpson, editors, *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, volume 216 of *LIPIcs*, pages 28:1–28:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

Per Martin-Löf.
An intuitionistic theory of types: Predicative part.
In *Studies in Logic and the Foundations of Mathematics*, volume 80, pages 73–118. Elsevier, 1975.

Per Martin-Löf.
Intuitionistic type theory, volume 1 of *Studies in Proof Theory*.
Bibliopolis, 1984.