Eliminating dependent size-change termination

Lide Grotenhuis and Daniël Otten
ILLC, University of Amsterdam

Minimalistic vs expressive theories

When designing a theory, there is a tension between making it:

- Minimalistic: easy to understand and model externally.
- Expressive: easy to read and write proofs internally.

We want both! And we can, by proving equivalences.

Today: inductive elimination principles. An *indexed inductive type* is specified by its *constructors*. For its *destructors*, there are choices:

- Minimalistic: the primitive eliminators, giving one-step induction:
  - a case distinction, with one clause for each constructor, using
  - one induction hypothesis for every direct predecessor.
- Expressive: general mutually recursive (dependent) functions:
  - pattern matching with nested constructors and unified indices, and
  - recursive calls satisfying a termination criterion.

The expressive rules should be interpreted in the minimalistic rules.

Eliminating pattern matching

Proof assistants allow us to define:

$$\begin{aligned}&\mathsf{distr} : (A + B) \times C \to (A \times C) + (B \times C),\\ &\mathsf{distr}\,(\mathsf{in}_0\,x, z) \stackrel{\mathsf{def}}{=} \mathsf{in}_0\,(x, z),\\ &\mathsf{distr}\,(\mathsf{in}_1\,y, z) \stackrel{\mathsf{def}}{=} \mathsf{in}_1\,(y, z).\end{aligned}$$

(nested constructors)

$$\begin{aligned}&\mathsf{trans} : \Pi_{x,y,z:A}\,(x = y) \to (y = z) \to (x = z),\\ &\mathsf{trans}\,x\,x\,x\,\mathsf{refl}_x\,\mathsf{refl}_x \stackrel{\mathsf{def}}{=} \mathsf{refl}_x,\end{aligned}$$

(unified indices)

Is the following allowed?

$$\begin{aligned}&\mathsf{uip} : \Pi_{x,y:A}\,\Pi_{p,q:(x=y)}\,(p = q),\\ &\mathsf{uip}\,x\,x\,\mathsf{refl}_x\,\mathsf{refl}_x \stackrel{\mathsf{def}}{=} \mathsf{refl}_x.\end{aligned}$$

Depends on the unification algorithm. Interpretations for both:

- Eliminating dependent pattern matching — Goguen–McBride–McKinna 2006
- Eliminating dependent pattern matching without K — Cockx–Devriese–Piessens 2016

Eliminating size-change termination

Termination is undecidable, so proof assistants check a criterion:

- One-step (Martin-Löf type theory): one recursive function where one argument is always a direct predecessor in every recursive call.
- Well-founded (`Lean`, `Rocq`): one recursive function where one argument always decreases in every recursive call.
- Size-change[1] (`Agda`, `Arend`, `Lambdapi`, `Idris`, `Isabelle/HOL`): mutually recursive functions where every infinite sequence of calls eventually has a chain that decreases infinitely often.

Well-founded induction can be interpreted in one-step induction.

Can size-change recursion be interpreted in well-founded induction?
Yes, using techniques from cyclic proof theory:

- Unravelling cyclic first-order arithmetic Leigh–Wehr 2026

[1] This condition does not require sized types.

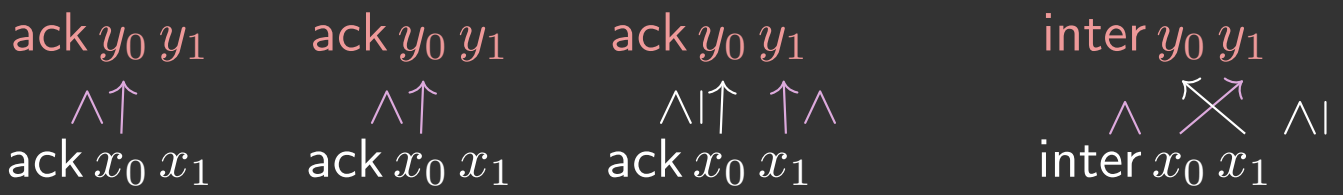
Examples

Consider the Ackermann and interleave functions:

$$\begin{aligned}
&\mathsf{ack} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}, \\
&\mathsf{ack}\, 0\, x_1 \stackrel{\mathsf{def}}{=} \mathsf{suc}\, x_1, \\
&\mathsf{ack}\, (\mathsf{suc}\, x_0')\, 0 \stackrel{\mathsf{def}}{=} \mathsf{ack}\, x_0'\, 1, \\
&\mathsf{ack}\, (\mathsf{suc}\, x_0')\, (\mathsf{suc}\, x_1') \stackrel{\mathsf{def}}{=} \mathsf{ack}\, x_0'\, (\mathsf{ack}\, (\mathsf{suc}\, x_0')\, x_1').
\end{aligned}$$

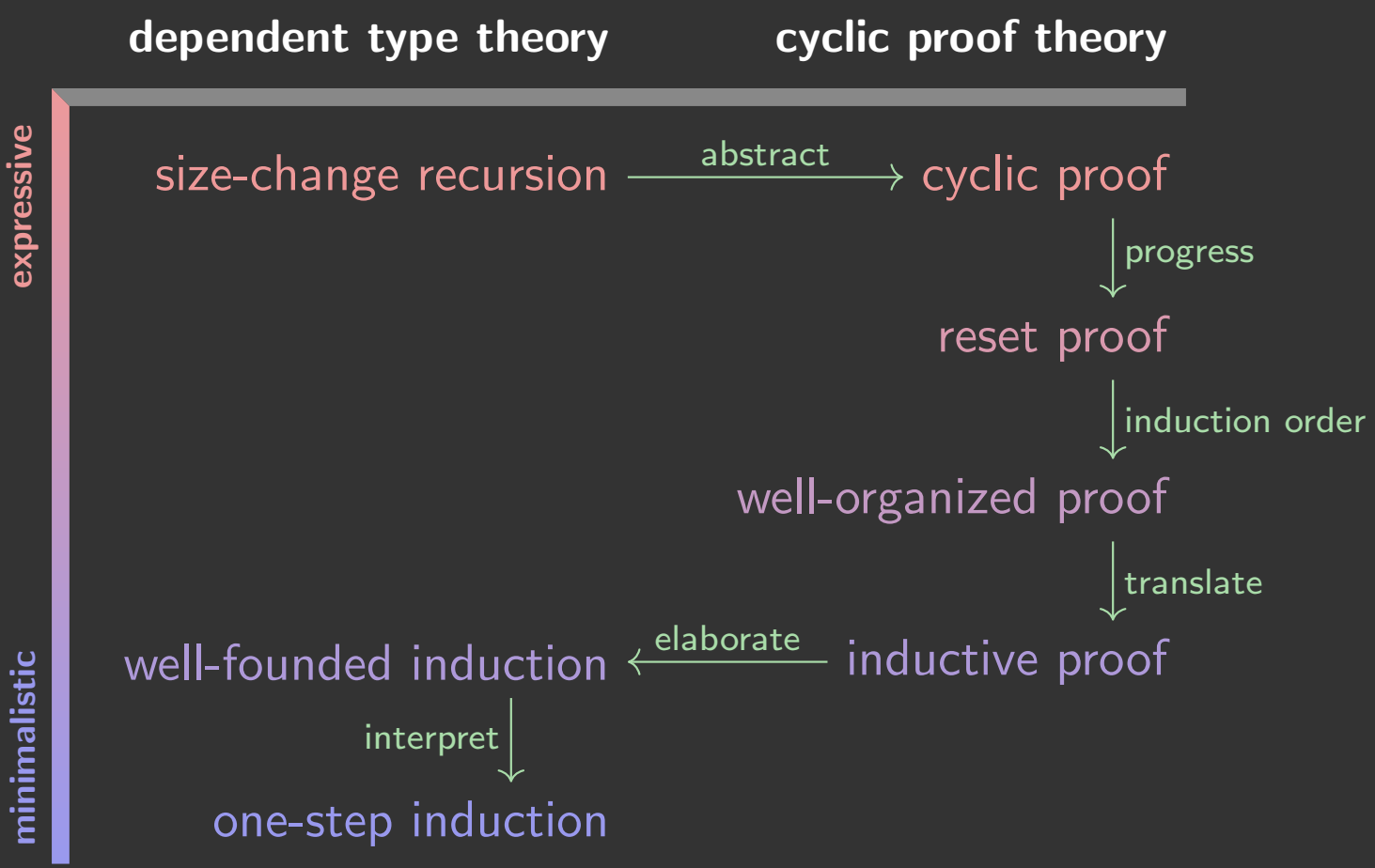
$$\begin{aligned}
&\mathsf{inter} : \mathsf{List}\, \mathbb{N} \to \mathsf{List}\, \mathbb{N} \to \mathsf{List}\, \mathbb{N}, \\
&\mathsf{inter}\, \mathsf{nil}\, x_1 \stackrel{\mathsf{def}}{=} x_1, \\
&\mathsf{inter}\, (n :: x_0')\, x_1 \stackrel{\mathsf{def}}{=} n :: \mathsf{inter}\, x_1\, x_0'.
\end{aligned}$$

Every recursive call has a size-change graph from the original inputs $\overline{x}$ to the recursive inputs $\overline{y}$: $>$ for a strict subterm, $\geq$ for an equal term.

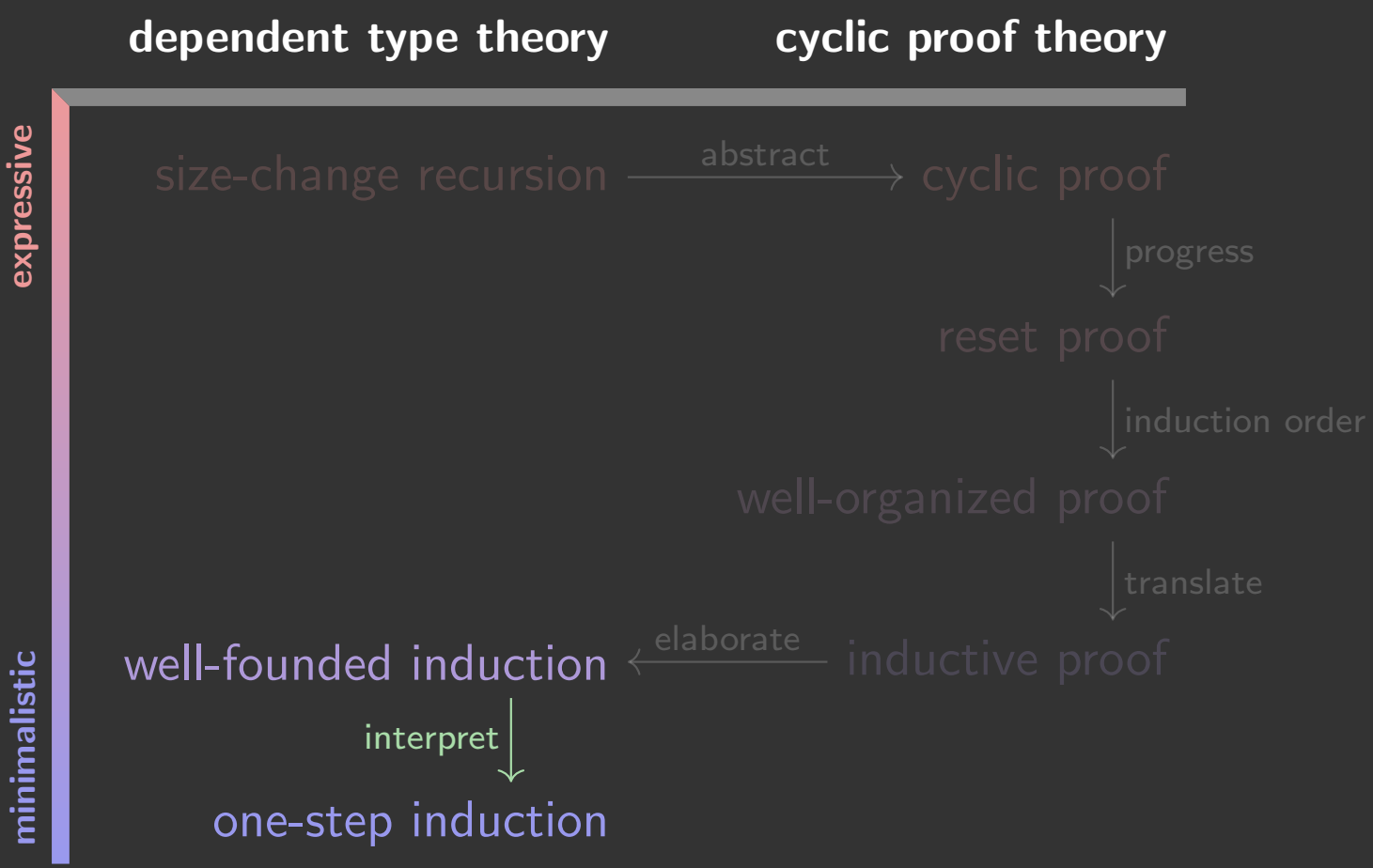


Both are size-change terminating (any infinite composition of these graphs has an infinitely decreasing chain), but not well-founded recursive.

Overview


dependent type theory
cyclic proof theory
expressive
minimalistic
size-change recursion
abstract
cyclic proof
progress
reset proof
induction order
well-organized proof
translate
elaborate
well-founded induction
inductive proof
interpret
one-step induction

Overview


dependent type theory
cyclic proof theory
expressive
minimalistic
size-change recursion
abstract
cyclic proof
progress
reset proof
induction order
well-organized proof
translate
inductive proof
elaborate
well-founded induction
interpret
one-step induction

One-step induction

As an example, W-types have *formation* and *introduction* rules:

$$\frac{\Gamma \vdash A : \mathsf{Type} \qquad \Gamma, x : A \vdash B : \mathsf{Type}}{\Gamma \vdash \mathrm{W}_{x:A}\, B : \mathsf{Type}}\, \mathrm{W}_{x:A}\, B\text{-form},$$

$$\frac{}{\Gamma \vdash \mathsf{tree} : \Pi_{x:A}\, ((B \to \mathrm{W}_{x:A}\, B) \to \mathrm{W}_{x:A}\, B)}\, \mathrm{W}_{x:A}\, B\text{-intro},$$

We get the primitive *elimination* and *computation* rules:

$$\frac{\begin{array}{l}\Gamma \vdash t : \mathrm{W}_{x:A}\, B \qquad \Gamma, w : \mathrm{W}_{x:A}\, B \vdash P : \mathsf{Type} \\ \Gamma, x : A, z : B \to \mathrm{W}_{x:A}\, B, h : \Pi_{y:B}\, P[z\, y] \vdash p : P[\mathsf{tree}\, x\, z]\end{array}}{\Gamma \vdash t.\mathsf{ind}_{w \mapsto P}\, \{\mathsf{tree}\, x\, z, h \mapsto p\} : P[t]}\, \mathrm{W}_{x:A}\, B\text{-elim},$$

$$\frac{}{(\mathsf{tree}\, x\, z).\mathsf{ind}_{w \mapsto P}\, \{\cdots\} \equiv p[x, z, \lambda_{y:B}\, (z\, y).\mathsf{ind}_{w \mapsto P}\, \{\cdots\}]}\, \mathrm{W}_{x:A}\, B\text{-comp}.$$

This allows case distinction and one-step induction: the induction hypothesis h is about the direct predecessors $z\, y$ only.

Structural order

The definitional structural order on terms of $\mathrm{W}_{x:A}\, B$ is given by:

$$\frac{}{z\,y \lessdot \mathsf{tree}\,x\,z}\ \lessdot\text{-base}, \qquad \frac{w' \lessdot z\,y}{w' \lessdot \mathsf{tree}\,x\,z}\ \lessdot\text{-step}. \qquad \frac{\begin{array}{l} w''' \equiv w'' \\ w'' \lessdot w' \\ w' \equiv w \end{array}}{w''' \lessdot w}\ \lessdot\text{-clo}.$$

To state well-founded induction, we need to state that P holds below w.

Previous work uses large elimination:

$$\mathsf{Below}_{w \mapsto P}[w] \stackrel{\mathsf{def}}{=} w.\mathsf{ind}_{w' \mapsto \mathcal{U}}\,\{\mathsf{tree}\,x\,z, H \mapsto \Pi_{y:B}\,(H\,y \times P[z\,y])\}.$$

Instead, we use a type $<$ for the propositional structural order:

$$\mathsf{Below}_{w \mapsto P}[w] \stackrel{\mathsf{def}}{=} \Pi_{w',q:w'<w}\,P[w'],$$

where $< : \mathrm{W}_{x:A}\, B \to \mathrm{W}_{x:A}\, B \to \mathsf{Type}$ is the indexed inductive type:

$$\mathsf{base} : \Pi_{x,z,y}\,(z\,y < \mathsf{tree}\,x\,z), \quad \mathsf{step} : \Pi_{x,z,y,w'}\,(w' < z\,y \to w' < \mathsf{tree}\,x\,z).$$

This needs no large elimination and is easier to compose.

Well-founded induction and composition

If $t' \ll t$, then we write $\mathsf{def}\text{-}<_t^{t'} : t' < t$ for the canonical term.

We can derive well-founded induction from one-step induction:

$$\frac{\begin{array}{l}\Gamma \vdash t : \mathrm{W}_{x:A}\, B \qquad \Gamma, w : \mathrm{W}_{x:A}\, B \vdash P : \mathsf{Type} \\ \Gamma, w : \mathrm{W}_{x:A}\, B, h : \Pi_{w',q:w'<w}\, P[w'] \vdash p : P\end{array}}{\begin{array}{l}\Gamma \vdash t.\mathsf{ind}\text{-}<_{w\mapsto P}\, \{w, h \mapsto p\} : P[t] \\ \Gamma \vdash t.\mathsf{below}_{w\mapsto P}\, \{w, h \mapsto p\} : \Pi_{w',q:w'<t}\, P[w']\end{array}} <\text{-elim}^{\mathsf{wf}},$$

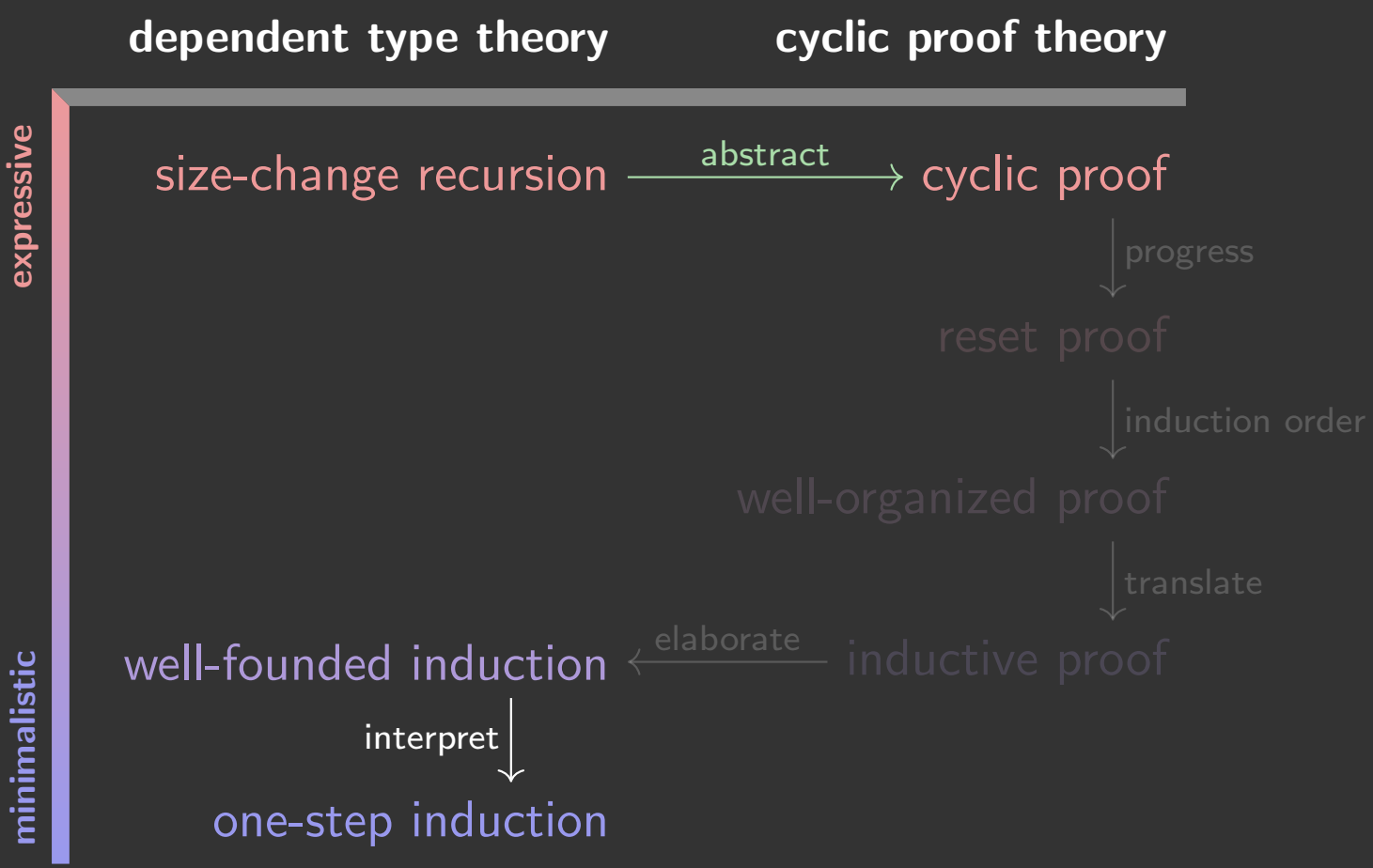
$$\frac{}{\begin{array}{l}t.\mathsf{ind}\text{-}<_{w\mapsto P}\, \{\cdots\} \equiv p[t, t.\mathsf{below}_{w\mapsto P}\, \{\cdots\}] \\ t.\mathsf{below}_{w\mapsto P}\, \{\cdots\}\, t'\, \mathsf{def}\text{-}<_t^{t'} \equiv t'.\mathsf{ind}\text{-}<_{w\mapsto P}\, \{\cdots\}\end{array}} <\text{-comp}^{\mathsf{wf}}.$$

The h is now about all structurally smaller w'. We also derive:

$$\frac{\Gamma \vdash q' : t'' < t' \qquad \Gamma \vdash q : t' < t}{\Gamma \vdash q' \cdot q : t'' < t} <\text{-elim}^{\mathsf{trans}},$$

$$\frac{}{\mathsf{def}\text{-}<_{t'}^{t''} \cdot \mathsf{def}\text{-}<_t^{t'} \equiv \mathsf{def}\text{-}<_t^{t''}} <\text{-comp}^{\mathsf{trans}}.$$

Overview


dependent type theory
cyclic proof theory
expressive
minimalistic
size-change recursion
abstract
cyclic proof
progress
reset proof
induction order
well-organized proof
translate
elaborate
well-founded induction
inductive proof
interpret
one-step induction

Recursive functions as cyclic proofs

Curry–Howard extends to this setting:

recursive function	cyclic proof
recursive call	cycle
size-change termination	global trace condition

For mutually recursive functions $\overline{f}$, we define a cyclic proof system $\mathcal{C}^{\overline{f}}$:

- judgment $f_k\,\overline{x}\downarrow$ for every function $f_k : \Pi_{\overline{x:\mathsf{Ind}_k}}\, D_k[\overline{x}]$ and variables $\overline{x}$;
- one rule for every function:

$$\frac{f_l\,\overline{y}\downarrow \qquad \text{for every recursive call to } f_l \text{ in } f_k}{f_k\,\overline{x}\downarrow}\, f_k.$$

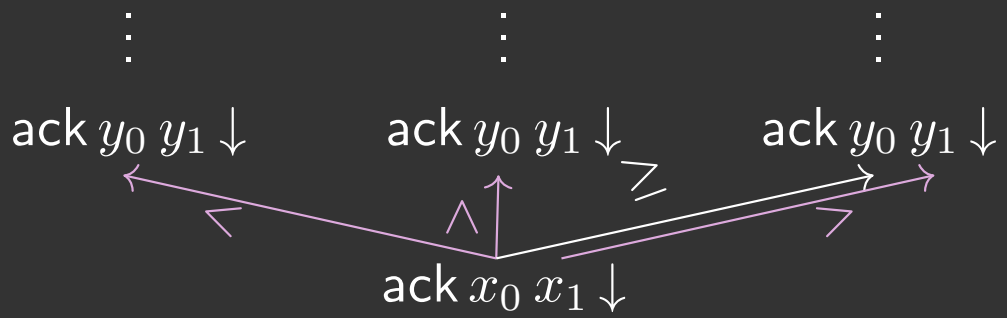
Each $f_k\,\overline{x}\downarrow$ has a unique possibly infinite derivation (up to renaming), and it is considered a proof iff for every infinite branch the compositions of the size-change graphs have an infinitely decreasing chain.

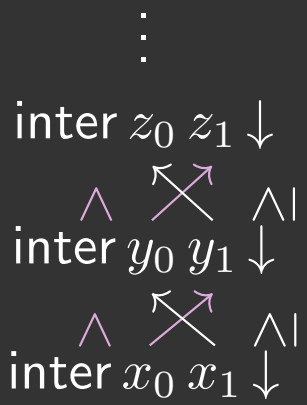
Examples

For Ackermann and interleave we get rules:

$$\frac{\mathsf{ack}\, y_0\, y_1 \downarrow \quad \mathsf{ack}\, y_0\, y_1 \downarrow \quad \mathsf{ack}\, y_0\, y_1 \downarrow}{\mathsf{ack}\, x_0\, x_1 \downarrow}\ \mathsf{ack}, \qquad \frac{\mathsf{inter}\, y_0\, y_1 \downarrow}{\mathsf{inter}\, x_0\, x_1 \downarrow}\ \mathsf{inter}.$$

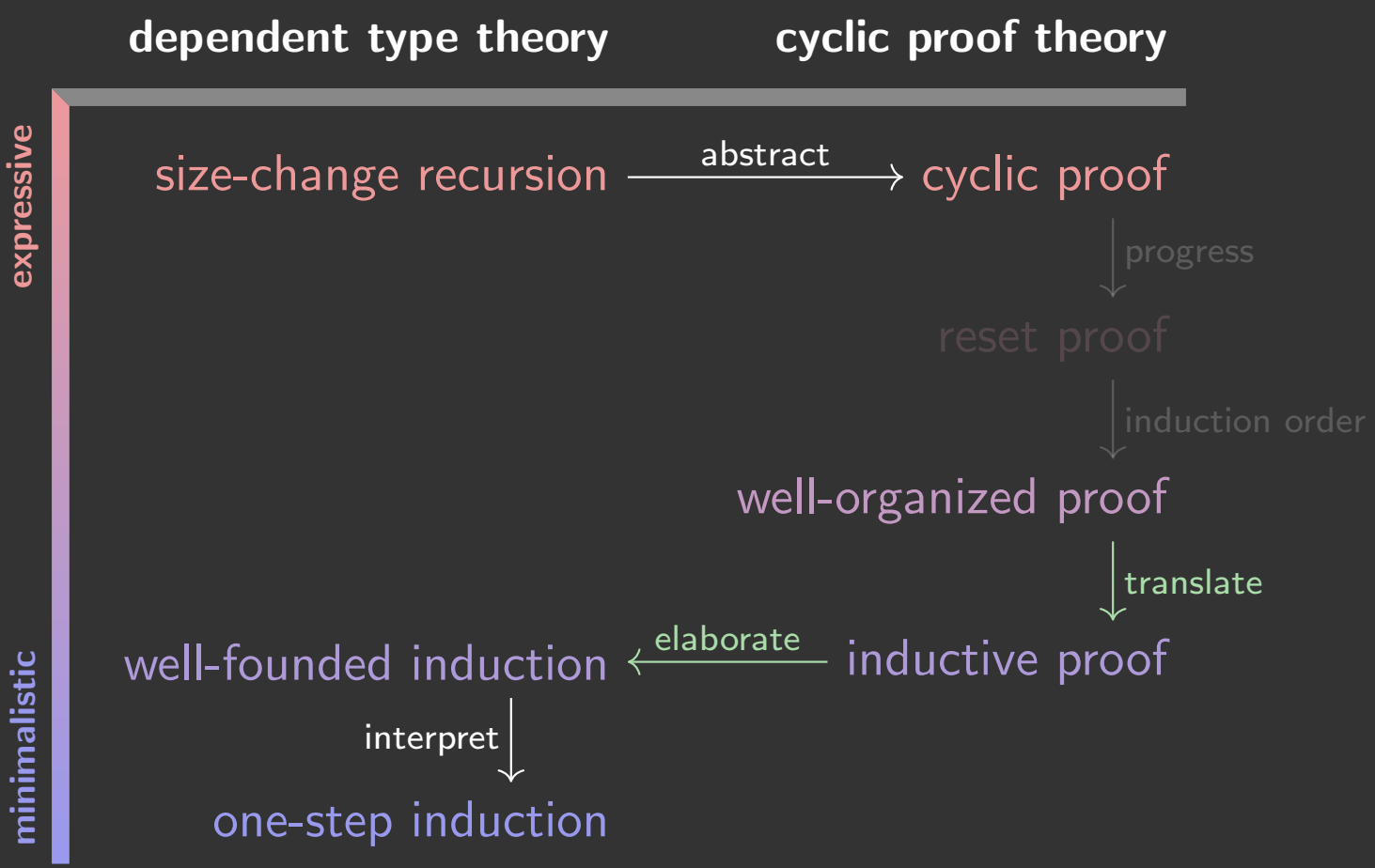
And the infinite derivations are:





Every branch has an infinitely decreasing chain.

Overview


dependent type theory
cyclic proof theory
expressive
minimalistic
size-change recursion
abstract
cyclic proof
progress
reset proof
induction order
well-organized proof
translate
elaborate
well-founded induction
inductive proof
interpret
one-step induction

The inductive system $\mathcal{C}^{\overline{f}}_{\mathsf{ind}}$

For any cyclic proof system $\mathcal{C}$, we define a (non-cyclic) proof system $\mathcal{C}_{\mathsf{ind}}$, as a multi-sorted intuitionistic predicate logic. The formulas of $\mathcal{C}^{\overline{f}}_{\mathsf{ind}}$ are:

$$\varphi, \psi ::= y <_{\mathsf{Ind}} x \mid f_k\, \overline{x}{\downarrow} \mid x =_{\mathsf{Ind}} y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi \to \psi \mid \exists_{x:\mathsf{Ind}}\, \varphi \mid \forall_{x:\mathsf{Ind}}\, \varphi$$

Judgments are sequents $\mathcal{X}; \Gamma \vdash \delta$. We have the intuitionistic rules, and:

$$\frac{\mathcal{X}, x : \mathsf{Ind}; \Gamma, \forall_{\mathcal{X}, x'}\, (x' < x \to (\Gamma \to \delta)[x']) \vdash \delta}{\mathcal{X}, x : \mathsf{Ind}; \Gamma \vdash \delta}\ <\text{-wf}, \qquad \frac{\mathcal{X}; \Gamma \vdash x < y \quad \mathcal{X}; \Gamma \vdash y < z}{\mathcal{X}; \Gamma \vdash x < z}\ <\text{-trans},$$

and for every function $f_k : \Pi_{\overline{x:\mathsf{Ind}_k}}\, D_k[\overline{x}]$ a rule:

$$\frac{\mathcal{X}, \overline{x : \mathsf{Ind}_k}, \overline{y : \mathsf{Ind}_l}; \Gamma, \overline{x} \gtrsim \overline{y} \vdash f_l\, \overline{y}{\downarrow} \qquad \text{for every recursive call to } f_l \text{ in } f_k}{\mathcal{X}, \overline{x : \mathsf{Ind}_k}; \Gamma \vdash f_k\, \overline{x}{\downarrow}}\ f.$$

Here $\overline{x} \gtrsim \overline{y}$ contains $x_i > y_j$ or $x_i \geq y_j$ iff this occurs in the size-change graph from f_k to f_l, where $x \geq y$ abbreviates $(x > y) \vee (x = y)$.

Annotating the infinite proof

Every infinite proof in $\mathcal{C}^{\overline{f}}$ becomes an infinite derivation in $\mathcal{C}^{\overline{f}}_{\mathsf{ind}}$:

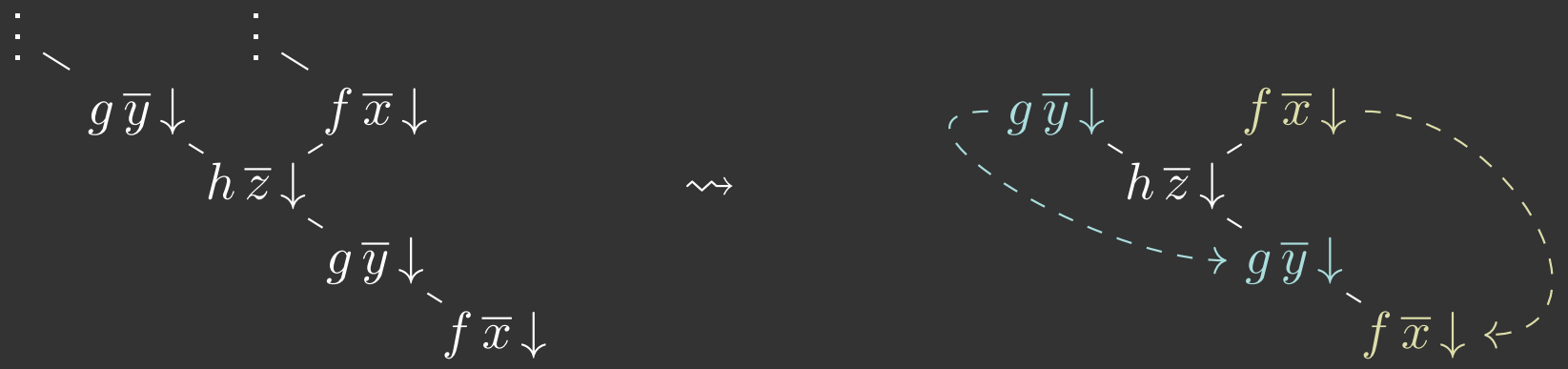
$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{f_3\,\overline{x}_3\downarrow}}{f_2\,\overline{x}_2\downarrow}\,f_2}{f_1\,\overline{x}_1\downarrow}\,f_1}{f_0\,\overline{x}_0\downarrow}\,f_0 \quad\rightsquigarrow\quad \dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\overline{x_0},\overline{x_1},\overline{x_2},\overline{x_3};\ \overline{x}_0 \gtrsim \overline{x}_1, \overline{x}_1 \gtrsim \overline{x}_2, \overline{x}_2 \gtrsim \overline{x}_3 \vdash f_3\,\overline{x}_3\downarrow}}{\overline{x_0},\overline{x_1},\overline{x_2};\quad \overline{x}_0 \gtrsim \overline{x}_1, \overline{x}_1 \gtrsim \overline{x}_2 \vdash f_2\,\overline{x}_2\downarrow}\,f_2}{\overline{x_0},\overline{x_1};\quad \overline{x}_0 \gtrsim \overline{x}_1 \vdash f_1\,\overline{x}_1\downarrow}\,f_1}{\overline{x_0} \vdash f_0\,\overline{x}_0\downarrow}\,f_0$$

We want a finite proof: cut the infinite branches short by introducing an induction hypothesis below, and using it above.

A proof in $\mathcal{C}^{\overline{f}}_{\mathsf{ind}}$ can be elaborated into a term.

The main idea

We represent the infinite proof of $f\,\overline{x}\!\downarrow$ as a finite tree with back-edges:



The infinite proof has many such cyclic representations.

Idea: pick a particularly nice (well-organized) one, and for back-edges:

- introduce an induction hypothesis below and use it above.

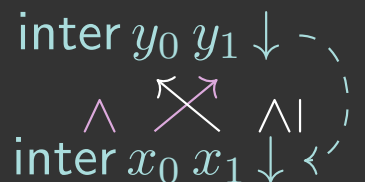
What makes a cyclic representation well-organized? We need:

- progress: to use an induction hypothesis, some input must decrease;
- preservation: inner inductions must preserve outer inductions;
- relevance: we should only keep assumptions that are still relevant; stronger statements give stronger induction hypotheses.

Progress: interleave

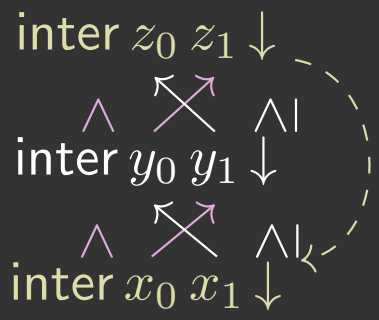
The proof of inter $x_0\, x_1 \downarrow$ has a cyclic representation:

$\mathsf{inter} : \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N},$
$\mathsf{inter}\,\mathsf{nil}\,x_1 \stackrel{\mathsf{def}}{=} x_1,$
$\mathsf{inter}\,(n :: x_0')\,x_1 \stackrel{\mathsf{def}}{=} n :: \mathsf{inter}\,x_1\,x_0'.$



But no input decreases. So we unfold, which corresponds to inlining:

$\mathsf{inter} : \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N},$
$\mathsf{inter}\,\mathsf{nil}\,x_1 \stackrel{\mathsf{def}}{=} x_1,$
$\mathsf{inter}\,(n :: x_0')\,\mathsf{nil} \stackrel{\mathsf{def}}{=} n :: x_0',$
$\mathsf{inter}\,(n :: x_0')\,(m :: x_1') \stackrel{\mathsf{def}}{=} n :: m :: \mathsf{inter}\,x_0'\,x_1'.$



Now $z_0 < x_0$, so we can use an induction hypothesis for x_0.

Progress: interleave

Following the cycle, we get a finite $\mathcal{C}^{\text{inter}}_{\text{ind}}$ proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\text{use } H_0[x_0] \text{ on } x_0' \overset{\text{def}}{=} z_0 \text{ and } x_1' \overset{\text{def}}{=} z_1)}{\overline{x}, \overline{y}, \overline{z}; H_0[x_0],\ x_0 > y_1,\ x_1 \geq y_0,\ y_0 > z_1,\ y_1 \geq z_0 \vdash \text{inter}\, z_0\, z_1 \downarrow}\ \text{inter}}{\overline{x}, \overline{y}; H_0[x_0],\ x_0 > y_1,\ x_1 \geq y_0 \vdash \text{inter}\, y_0\, y_1 \downarrow}\ \text{inter}}{\overline{x}; H_0[x_0] \vdash \text{inter}\, x_0\, x_1 \downarrow}\ <\text{-wf}_{x_0},}{\overline{x} \vdash \text{inter}\, x_0\, x_1 \downarrow}$$

where $H_0[x_0] \overset{\text{def}}{=} \forall_{x_0', x_1'}\ (x_0' < x_0 \to \text{inter}\, x_0'\, x_1' \downarrow)$.

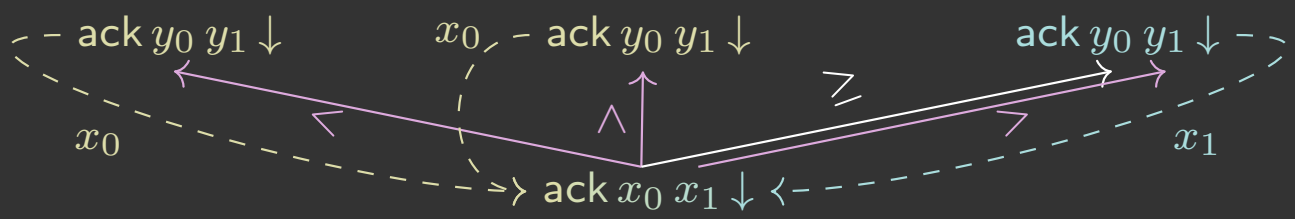
Elaborating this $\mathcal{C}^{\text{inter}}_{\text{ind}}$ proof to type theory gives a term:

$$\text{inter}_{\text{ind}} \overset{\text{def}}{=} \lambda_{x_0, x_1}\ x_0.\text{ind-}<\{x_0, h_0 : \Pi_{x_0', x_1'}\ (x_0' < x_0 \to \text{List}\,\mathbb{N}) \mapsto \lambda_{x_1}\ \text{inner}\}\, x_1,$$

$$\text{inner} \overset{\text{def}}{=} x_0.\text{case}\left\{\begin{array}{l} \text{nil} \mapsto \lambda_{h_0}\ x_1, \\ n :: x_0' \mapsto \lambda_{h_0}\ x_1.\text{case}\left\{\begin{array}{l} \text{nil} \mapsto n :: x_0' \\ m :: x_1' \mapsto n :: m :: h_0\, x_0'\, x_1'\ \text{def-}< \end{array}\right. \end{array}\right\}\ h_0.$$

Preservation: Ackermann

The proof of $\mathsf{ack}\, x_0\, x_1 \downarrow$ has a cyclic representation:



Applying induction twice, we get a finite proof in $\mathcal{C}^{\overline{f}}_{\mathsf{ind}}$:

$$\dfrac{\dfrac{\dfrac{(\text{use } H_0[x_0] \text{ on } x_i' \overset{\text{def}}{=} y_i)}{\overline{x}, \overline{y}; x_0 > y_0 \vdash \mathsf{ack}\, y_0\, y_1 \downarrow}\,(2\times) \quad \dfrac{(\text{use } H_1[x_1] \text{ on } x_i' \overset{\text{def}}{=} y_i)}{\overline{x}, \overline{y}; x_0 \geq y_0, x_1 > y_1 \vdash \mathsf{ack}\, y_0\, y_1 \downarrow}}{\overline{x}; H_0[x_0], H_1[x_1] \vdash \mathsf{ack}\, x_0\, x_1 \downarrow}\,\mathsf{ack}}{\overline{x}; H_0[x_0] \vdash \mathsf{ack}\, x_0\, x_1 \downarrow}\,<\text{-wf}_{x_1}}{\overline{x} \vdash \mathsf{ack}\, x_0\, x_1 \downarrow}\,<\text{-wf}_{x_0},$$

$$H_0[x_0] \overset{\text{def}}{=} \forall_{x_0', x_1'}\, (x_0' < x_0 \to \mathsf{ack}\, x_0'\, x_1' \downarrow),$$

$$H_1[x_1] \overset{\text{def}}{=} \forall_{x_0', x_1'}\, (x_1' < x_1 \to H_0[x_0'] \to \mathsf{ack}\, x_0'\, x_1' \downarrow).$$

Note that $H_0[x_0]$ and $x_0' \leq x_0$ give $H_0[x_0']$. It would <u>not</u> work if we introduce the induction hypotheses in the other order.

Preservation: Ackermann

Recall the recursive definition of the Ackermann function:

$$\begin{aligned}
&\mathsf{ack} : \mathbb{N} \to \mathbb{N} \to \mathbb{N},\\
&\mathsf{ack}\,0\,x_1 \overset{\mathsf{def}}{=} \mathsf{suc}\,x_1,\\
&\mathsf{ack}\,(\mathsf{suc}\,x_0')\,0 \overset{\mathsf{def}}{=} \mathsf{ack}\,x_0'\,1,\\
&\mathsf{ack}\,(\mathsf{suc}\,x_0')\,(\mathsf{suc}\,x_1') \overset{\mathsf{def}}{=} \mathsf{ack}\,x_0'\,(\mathsf{ack}\,(\mathsf{suc}\,x_0')\,x_1').
\end{aligned}$$

Elaborating the finite $\mathcal{C}^{\mathsf{ack}}_{\mathsf{ind}}$ proof gives the term:

$$\mathsf{ack}_{\mathsf{ind}} \overset{\mathsf{def}}{=} \lambda_{x_0,x_1}\, x_0.\mathsf{ind}\text{-}{<}\,\{x_0, h_0 \mapsto \lambda_{x_1}\, x_1.\mathsf{ind}\text{-}{<}\,\{x_1, h_1 \mapsto \lambda_{x_0,h_0}\, \mathsf{inner}\}\, x_0\, h_0\}\, x_1,$$

$$\mathsf{inner} \overset{\mathsf{def}}{=} x_0.\mathsf{case}\left\{\begin{array}{l}\mathsf{zero} \mapsto \lambda_{h_0}\, \mathsf{suc}\, x_1,\\ \mathsf{suc} x_0' \mapsto \lambda_{h_0}\, x_1.\mathsf{case}\left\{\begin{array}{l}\mathsf{zero} \mapsto \lambda_{h_1}\, h_0 x_0' 1 \mathsf{def}\text{-}{<}\\ \mathsf{suc} x_1' \mapsto \lambda_{h_1}\, h_0 x_0' (h_1 (\mathsf{suc} x_0') x_1' \mathsf{def}\text{-}{<} h_0) \mathsf{def}\text{-}{<}\end{array}\right\} h_1\end{array}\right\} h_0.$$

Relevance: forgetting

The more we assume, the weaker a new induction hypothesis becomes:

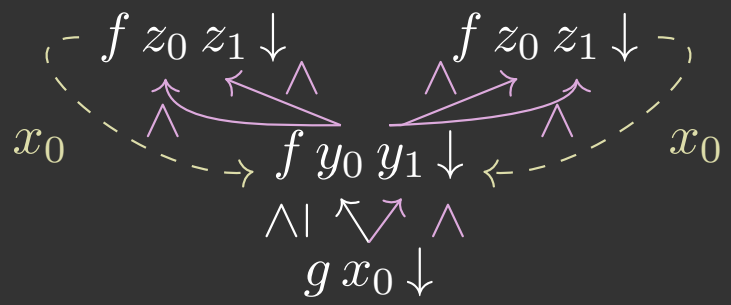


- Left, we can forget the hyp for x_0 before introducing a hyp for x_1.
- Right, we cannot forget: the back-edge for x_1 has to preserve x_0.

In general, we only want to keep assumptions that are still relevant.

Relevance: remembering

Besides induction hypotheses, it can be crucial to remember *inequalities*:



Induction on just y_0 or just y_1 does not work. If we remember x_0, $y_0 \leq x_0$, and $y_1 < x_0$ at $f\ y_0\ y_1\downarrow$, we can use:

$$H_0[x_0] \stackrel{\text{def}}{=} \forall_{x_0', y_0', y_1'}\ (x_0' < x_0 \to y_0' \leq x_0' \to y_1' < x_0' \to f\ y_0'\ y_1'\downarrow).$$

Previous work by Leigh–Wehr does induction on $\max y_0\ y_1$: this does not exist for non-linear orders (trees, W-types), and is bad for *computation*.

Well-organized cyclic representation

A well-organized cyclic representation picks for every node n with label f_n a $\mathcal{C}^{\overline{f}}_{\mathsf{ind}}$-sequent $\mathcal{X}_n; \mathcal{Q}_n, \mathcal{H}_n \vdash f_n\, \overline{x}_n\!\downarrow$, consisting of:

- a variable context $\mathcal{X}_n$ of relevant variables (past and current),
- a formula context $\mathcal{Q}_n$ of relevant inequalities,
- a formula context $\mathcal{H}_n$ of relevant induction hypotheses,

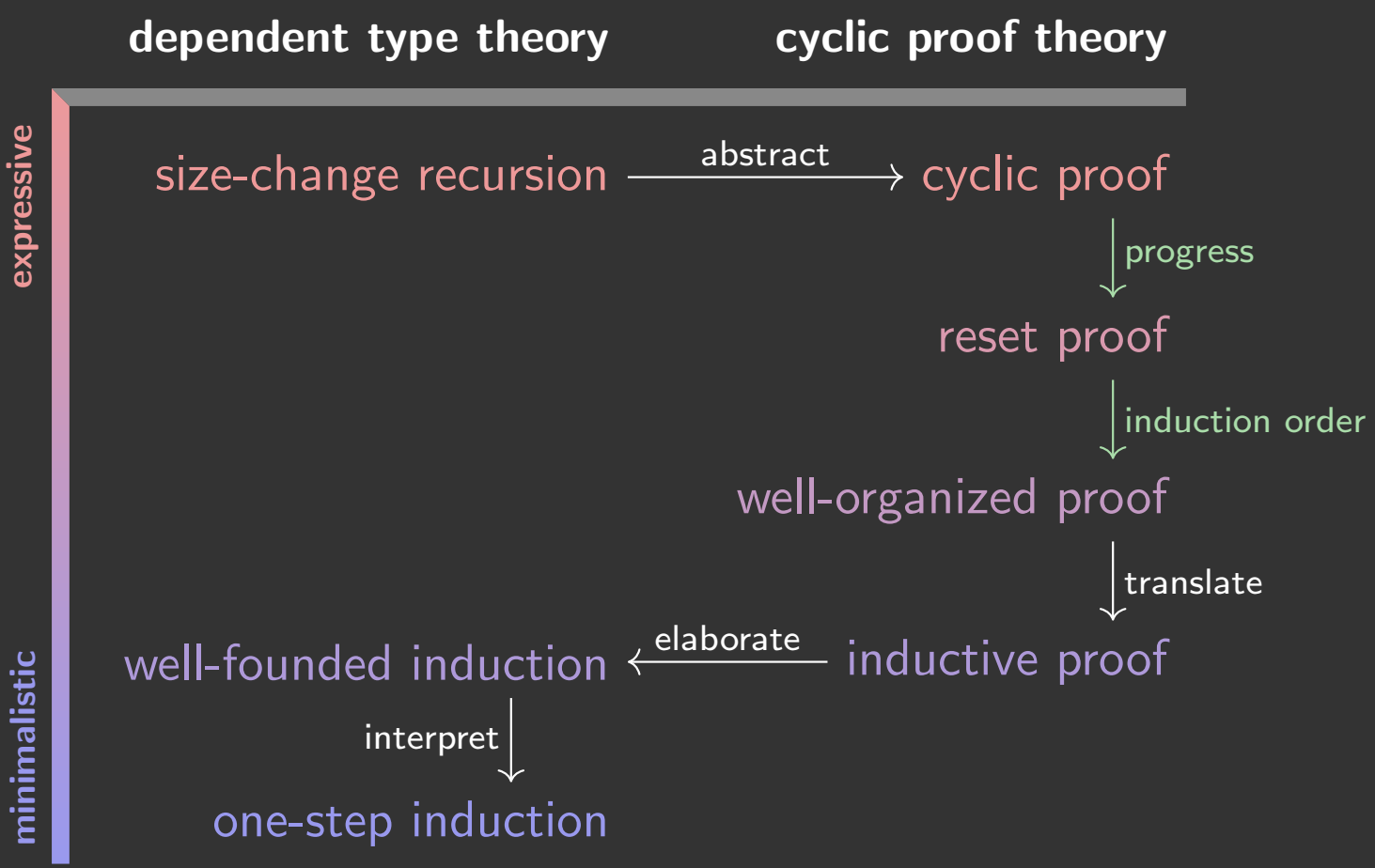
where higher nodes may *forget* data from lower nodes.

For every back-edge from b down to s we have a progress variable x_{prog_b}, a substitution ρ_b sending variables in $\mathcal{X}_s$ to variables in $\mathcal{X}_b$, with:

1. progress: $\mathcal{X}_b; \mathcal{Q}_b, \mathcal{H}_b \vdash x_{\mathsf{prog}_b} > x_{\mathsf{prog}_b}\,[\rho_b]$,
2. preservation: $\mathcal{X}_b; \mathcal{Q}_b, \mathcal{H}_b \vdash \mathcal{Q}_s[\rho_b], \mathcal{H}_s[\rho_b]$,
3. input: $\mathcal{X}_b; \mathcal{Q}_b, \mathcal{H}_b \vdash \overline{x}_b = \overline{x}_s[\rho_b]$.

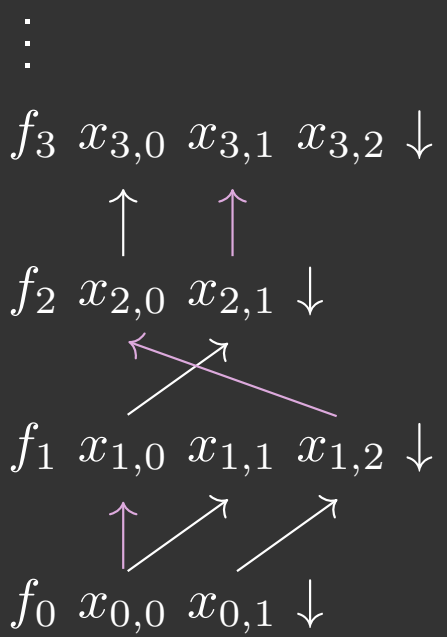
Does such a representation *always exist*? How do we *find* it?

Overview


dependent type theory
cyclic proof theory
expressive
minimalistic
size-change recursion
abstract
cyclic proof
progress
reset proof
induction order
well-organized proof
translate
elaborate
well-founded induction
inductive proof
interpret
one-step induction

Reset proofs

We use techniques from Leigh–Wehr and Jungteerapanich, based on Safra's determinization of stream automata. Given an infinite branch:



when can we introduce a back-edge? We track the history of the $\overline{x_n}$:

- remember: we remember some of the variables $x_{m,j} \geq x_{n,i}$: the oldest one, and some of the ones just after an $>$-edge;
- reset: when two variables are in the history of precisely the same $\overline{x_n}$, remember the oldest one, and forget variables younger than it.

When history repeats itself with progress, introduce a back-edge.

Induction order

Theorem

Every cyclic proof has a well-organized cyclic representation.

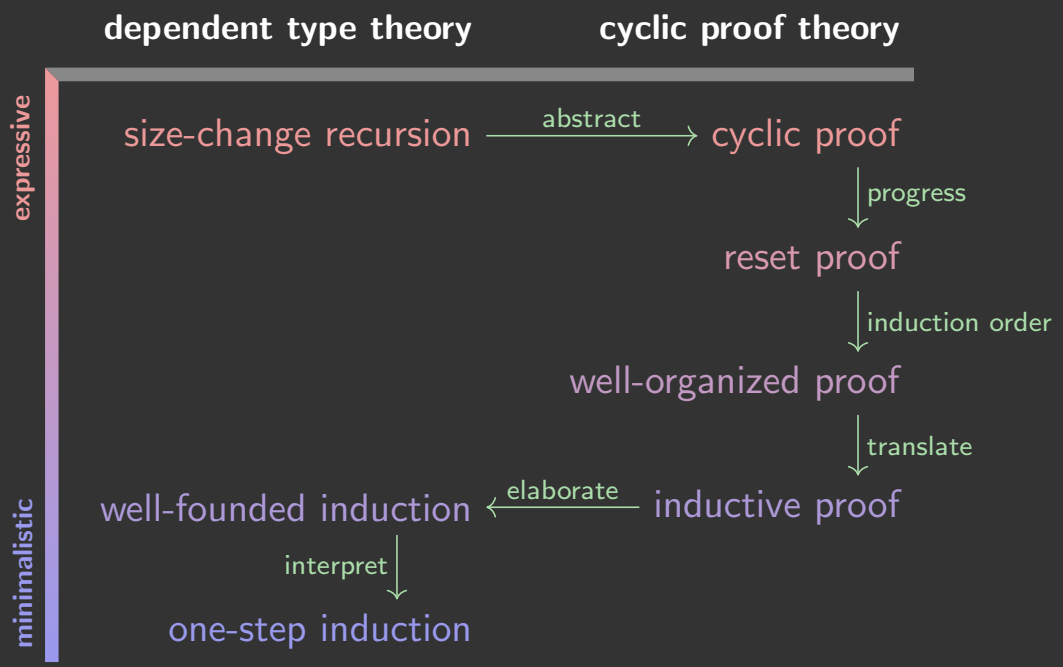
Proof sketch. For each $f_n\,\overline{x_n}\downarrow$, the relevant inequalities $\mathcal{Q}_n$ are the ones between two variables that are both remembered for some $x_{n,i}$, and the relevant induction hypotheses $\mathcal{H}_n$ are the ones that we need higher in the tree. The substitution comes from the fact that history repeats itself.

Inner induction does not yet respect outer induction, but following Sprenger–Dam, we can fix this by unfolding cycles. □

Main result

Theorem

All size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ have interpretations $\overline{f_{\mathsf{ind}}}$ in primitive eliminators (without axiom K or funext).



Main result

Theorem

All size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ have interpretations $\overline{f_{\mathsf{ind}}}$ in primitive eliminators (without axiom K or funext).

Proof sketch. For each function $f_k : \Pi_{\overline{x:\mathsf{Ind}_k}} D_k[\overline{x}]$ we take a well-organized cyclic representation of its unique infinite proof. With downwards induction, we define for each node n a term:

$$\mathcal{X}_n, \mathcal{Q}_n, \mathcal{H}_n \vdash d_n : D_{k_n}[\overline{x_n}]$$

We get $\overline{x : \mathsf{Ind}_k} \vdash d_{\mathsf{root}} : D_k[\overline{x}]$, and define $f_{\mathsf{ind},k} \stackrel{\mathsf{def}}{=} \lambda_{\overline{x}}\, d_{\mathsf{root}}$. □

What about computational behaviour? Do we have for every clause $f_k\, \overline{p} \stackrel{\mathsf{def}}{=} d[\overline{f\ a}]$ defining $f_k : \Pi_{\overline{x:\mathsf{Ind}_k}} D_k[\overline{x}]$ that $f_{\mathsf{ind},k}\, \overline{p} \equiv d[\overline{f_{\mathsf{ind}}\ a}]$?

Unfortunately not if the well-organized cyclic representation has more than one node that is not a leaf.

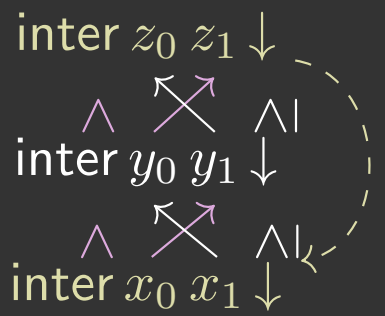
Open terms do not compute definitionally

Theorem

All size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ have interpretations $\overline{f_{\mathsf{ind}}}$ in primitive eliminators (without axiom K or funext).

We already see this for $\mathsf{inter} : \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N}$. To get a well-organized cyclic representation, we had to unfold/inline:

$$\mathsf{inter}\,\mathsf{nil}\,x_1 \stackrel{\mathsf{def}}{=} x_1,$$
$$\mathsf{inter}\,(n :: x_0')\,\mathsf{nil} \stackrel{\mathsf{def}}{=} n :: x_0',$$
$$\mathsf{inter}\,(n :: x_0')\,(m :: x_1') \stackrel{\mathsf{def}}{=} n :: m :: \mathsf{inter}\,x_0'\,x_1'.$$



But now $\mathsf{inter}_{\mathsf{ind}}\,(n :: x_0')\,x_1 \not\equiv n :: \mathsf{inter}_{\mathsf{ind}}\,x_1\,x_0'$: computation is stuck until we know that x_1 is nil or $m :: x_1'$.

Same for clauses like $f\,(\mathsf{suc}\,x_0')\,(\mathsf{suc}\,x_1') \stackrel{\mathsf{def}}{=} f\,x_0'\,x_0' + f\,x_1'\,x_1'$.
However, for functions like ack, that terminate due to a lexicographic order on inputs, the translation $\mathsf{ack}_{\mathsf{ind}}$ computes fully definitionally.

Closed terms compute definitionally

Theorem

All size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ have interpretations $\overline{f_{\mathsf{ind}}}$ in primitive eliminators (without axiom K or funext).

Proposition

If $f_k\,\overline{a}$ reduces to a normal form n in which $\overline{f}$ does not occur, then $f_{\mathsf{ind},k}\,\overline{a}$ also reduces to the normal form n.

Proof sketch. $f_{\mathsf{ind},k}$ corresponds to a well-organized cyclic representation. With downwards induction on the well-organized cyclic representation, we show that each term $\mathcal{X}_n, \mathcal{Q}_n, \mathcal{H}_n \vdash d_n : D[\overline{x_n}]$ computes under substitutions filling in def-$\lesssim$ for all $\mathcal{Q}_n$ and below for all $\mathcal{H}_n$.

Then use induction on the length of the reduction to show for any term $b[\overline{f\,\overline{a}}]$ that reduces to a term n without $\overline{f}$ that the term $b[\overline{f_{\mathsf{ind}}\,\overline{a}}]$ also reduces to n. □

Open terms compute propositionally

Theorem

All size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ have interpretations $\overline{f_{\mathsf{ind}}}$ in primitive eliminators (without axiom K or funext).

Proposition

If $f_k\,\overline{a}$ reduces to a normal form n in which $\overline{f}$ does not occur, then $f_{\mathsf{ind},k}\,\overline{a}$ also reduces to the normal form n.

Proposition

For every $f_k\,\overline{p} \stackrel{\mathsf{def}}{=} d[\overline{f\,\overline{a}}]$ of $f_k : \Pi_{\overline{x:\mathsf{Ind}_k}} D_k[\overline{x}]$ we have $f_{\mathsf{ind},k}\,\overline{p} = d[\overline{f_{\mathsf{ind}}\,\overline{a}}]$.

Proof sketch (under construction). For each f_k we define the graph as an indexed inductive type $F_k : \Pi_{\overline{x:\mathsf{Ind}_k}} (D_k[\overline{x}] \to \mathsf{Type})$, where each clause $f_k\,\overline{p} \stackrel{\mathsf{def}}{=} d[\overline{f\,\overline{a}}]$ gives a constructor clause $: \overline{F\,\overline{a}\,r} \to F_k\,\overline{p}\,d[\overline{r}]$. Then $F_k\,\overline{a}\,r \to F_k\,\overline{a}\,r' \to r = r'$, and with downwards induction on the cyclic representation, we show for $\mathcal{X}_n, \mathcal{Q}_n, \mathcal{H}_n \vdash d_n : D[\overline{x_n}]$ that $F_{k_n}\,\overline{x_n}\,d_n$. □

Open terms compute propositionally

Theorem

All size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ have interpretations $\overline{f_{\mathsf{ind}}}$ in primitive eliminators (without axiom K or funext).

Proposition

If $f_k\,\overline{a}$ reduces to a normal form n in which $\overline{f}$ does not occur, then $f_{\mathsf{ind},k}\,\overline{a}$ also reduces to the normal form n.

Proposition

For every $f_k\,\overline{p} \stackrel{\mathsf{def}}{=} d[\overline{f\,\overline{a}}]$ of $f_k : \Pi_{\overline{x:\mathsf{Ind}_k}} D_k[\overline{x}]$ we have $f_{\mathsf{ind},k}\,\overline{p} = d[\overline{f_{\mathsf{ind}}\,\overline{a}}]$.

So, in extensional type theory, adding size-change terminating mutually recursive $\overline{f : \Pi_{\overline{x:\mathsf{Ind}}} D[\overline{x}]}$ is conservative, but this remains open for intensional type theory.

We think this might be the best possible: in particular, that inter cannot be interpreted in primitive eliminators with full computational behaviour.

Preprint

We have a preprint covering the abstract cyclic proof theory:



`https://arxiv.org/abs/2602.12054`

A preprint covering the application to type theory is in the finishing stages, and will be uploaded soon under the same title as the talk.

For discussion

- We think our methods extend to indexed inductive types: combine with unification, define $\lll$ and $<$ on $\Sigma_{\overline{u:I}} \operatorname{Ind} \overline{u}$, use McBride's basic analysis, and Cockx's strong unifiers for both with and without K. How about inductive-inductive types and higher inductive types?
- What happens if we combine with coinduction and copatterns? Our abstract cyclic proof theory result can be applied to cyclic proof systems with coinductive sorts, although it only uses induction. Similar to sized types: (co)induction becomes induction on ordinals.
- Can any eliminator term interpret $\mathsf{inter} : \operatorname{List} \mathbb{N} \to \operatorname{List} \mathbb{N} \to \operatorname{List} \mathbb{N}$ with all definitional equalities? If not, is size-change termination consistent with univalence? This particular example should be fine in presheaf models; are there examples that are not?
- The result that any proof of an abstract cyclic proof system has a well-organized cyclic representation uses a classical metatheory. We have some ideas for making this constructive: instead of looking at all infinite branches, we should look at ultimately periodic branches. This is equivalent classically due to Ramsey's theorem.