Unravelling Abstract Cyclic Proofs into Proofs by Induction

Lide Grotenhuis and Daniël Otten
ILLC, University of Amsterdam

Cyclic proofs

Cyclic proof systems replace (co)induction rules with circular reasoning.

Example. Consider the language of arithmetic with axioms:

$$0 + x = x, \qquad \mathsf{suc}\, x + y = \mathsf{suc}(x + y).$$

We have a cyclic proof:

$$\dfrac{\dfrac{}{\mathsf{suc}\, x + 0 = \mathsf{suc}\,(x + 0)}\,\text{axiom} \qquad x + 0 = x}{\vdots}$$
$$\dfrac{\dfrac{}{0 + 0 = 0}\,\text{axiom} \qquad \mathsf{suc}\, x + 0 = \mathsf{suc}\, x}{x + 0 = x}\,\text{case}_x$$

Upshot: instead of guessing an induction hypothesis, we generate a proof until we find a repeat with progress.

Soundness of cyclic proofs

For a cyclic proof system, we need to specify which cycles are allowed:

- we want to be restrictive enough to be sound;
- we want to be admissive enough to be complete, and easy to use.

This is called the soundness condition.

A common condition is the global trace condition: for every infinite path we can eventually trace an object that makes progress infinitely often.

Example. For arithmetic:

- trace objects: variables,
- progress: passing through a case distinction.

In general, checking the global trace condition is PSPACE-complete.

Overview

To reconcile cyclic proofs with traditional proof theory, we want to transform them into into proofs by induction:

- So far, this has been done for various specific proof systems[1].
- We show how to do this in a general abstract way.
- In addition, we want to preserve the structure of the proof.

In particular:

- We start with an abstract cyclic proof system $\mathcal{C}$.
- We define an induced (non-cyclic) proof system $\mathcal{C}_{\text{ind}}$ that extends the rules of $\mathcal{C}$ with an intuitionistic induction principle.
- We show how any cyclic proof π in $\mathcal{C}$ can be transformed into a finite proof π_{ind} in $\mathcal{C}_{\text{ind}}$ with the same conclusion and structure.

By the Curry-Howard correspondence, this applies to recursive functions.

Our approach is greatly inspired by the PhD thesis of Dominik Wehr.

[1] arithmetic, modal/linear/first-order μ-calculus, inductive definitions, ...

Size-change graph

Definition (size-change graph)

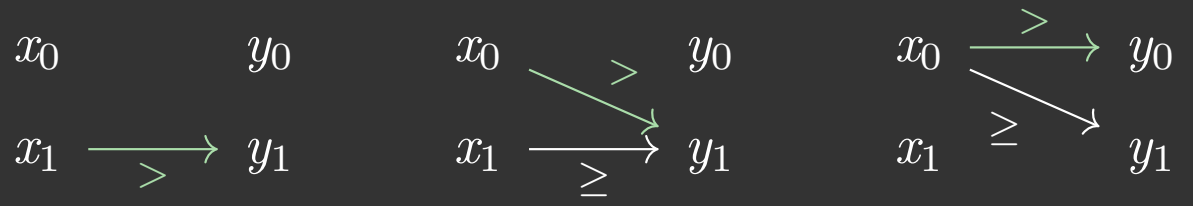
A *size-change graph* $G : X \to Y$ is a bipartite graph from a finite set X to a finite set Y where edges are labelled as either:

$$\geq \text{ (preserving)} \qquad \text{or} \qquad > \text{ (progressing)}.$$

A *trace* through a sequence of size-change graphs $(G_i : X_i \to X_{i+1})_i$ consists of a starting time $k \in \mathbb{N}$ and nodes $(x_i \in X_i)_{i \geq k}$ such that x_i and x_{i+1} are connected by an edge in G_i.

We call an infinite sequence *progressing* if there exists a trace where the connecting edge is progressing infinitely often.

Examples:



Cyclic proof system

Definition (cyclic proof system)

A *proof system* consists of:

- a set Judg *(the judgments)*, a set Rule *(the rule instances)*,
- for each rule $r \in \mathrm{Rule}$ an associated finite list of *premises* $\mathrm{Prem}(r) \in \mathrm{Judg}^*$ and a *conclusion* $\mathrm{concl}(r) \in \mathrm{Judg}$.

A *cyclic proof system* is a proof system with additional structure:

- for each judgment $A \in \mathrm{Judg}$ a number of *(trace) objects* $\mathrm{ob}(A) \in \mathbb{N}$,
- for each rule $r \in \mathrm{Rul}$ and each premise $\mathrm{Prem}(r)_i$ a *size-change graph*

$$\mathrm{Graph}(r)_i : \mathrm{ob}(\mathrm{concl}(r)) \to \mathrm{ob}(\mathrm{Prem}(r)_i).$$

A *derivation* is a (possibly infinite) tree where every node is labelled by a rule in a compatible way. A derivation is called a *proof* if it is *regular* (it only has finitely many subderivations) and for every infinite branch, the induced sequence of size-change graphs is progressing.

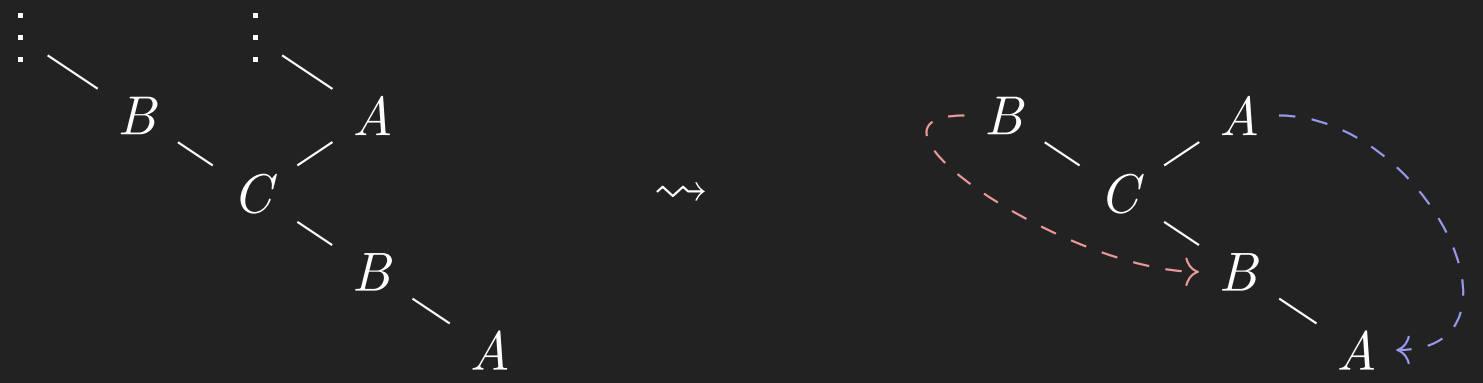
Cyclic representation

So, the proofs are possibly infinite trees. However, because they are regular we can always represent them in a finite way:

- As a finite derivation with back-edges. For a back-edge, we call the starting leaf the bud and the ending internal node the sprout.

We say that the infinite proof has a cyclic representation.

Example. The following infinite proof has a cyclic representation:

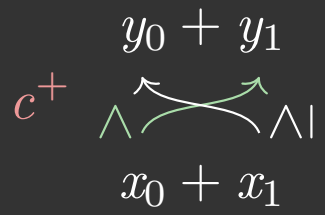


Recursive functions

Proof assistants only allow total functions, an admissive check is the size-change termination condition: for every infinite sequence of recursive calls, we can eventually trace an input that decreases infinite often.

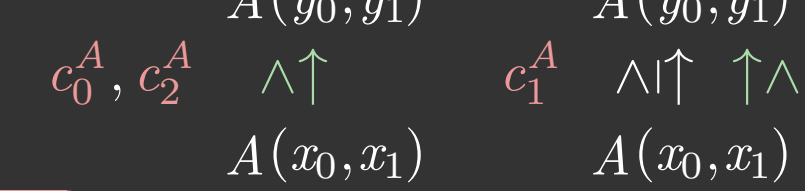
Consider addition with a swapped recursive call:

$$\begin{aligned} 0 + x_1 &:= x_1, \\ \mathsf{suc}(x_0') + x_1 &:= \mathsf{suc}(\overbrace{x_1 + x_0'}^{c^+}), \end{aligned}$$



Or the Ackermann function:

$$\begin{aligned} A(0, x_1) &:= \mathsf{suc}(x_1), \\ A(\mathsf{suc}(x_0'), 0) &:= \underbrace{A(x_0', 1)}_{c_0^A}, \\ A(\mathsf{suc}(x_0'), \mathsf{suc}(x_1')) &:= \underbrace{A(x_0', \overbrace{A(\mathsf{suc}(x_0'), x_1')}^{c_1^A})}_{c_2^A}. \end{aligned}$$



Upshot: much easier to write and read than recursive functions defined via the primitive elimination rules of dependent type theory.

Correspondence

Every set of mutually recursive functions induces a cyclic proof system:

- for every function f we have a judgment $\downarrow f$ (read: f *terminates*) with $\mathsf{ob}(\downarrow f) = \mathsf{ar}(f)$ and a rule

$$\frac{\downarrow g \qquad \text{for every } g \text{ called within } f}{\downarrow f}\, f,$$

  where for every recursive call the size-change graph is given by the relation between the input of f and g.

Note that every function has a unique derivation, and the function is size-change terminating iff this derivation is a proof.

This is an instance of the Curry-Howard correspondence:

Cyclic Proof	Recursive Function
Least Fixpoint Formula	Inductive Type
Greatest Fixpoint Formula	Coinductive Type
Cycle	Recursive Function Call
Soundness Condition	Termination Checking

Inductive proof system

Given a cyclic proof system $\mathcal{C}$ we define a (non-cyclic) proof system $\mathcal{C}_{\mathsf{ind}}$ with an explicit induction rules. The judgements of $\mathcal{C}_{\mathsf{ind}}$ are sequents $\gamma_0, \dots, \gamma_{n-1} \vdash \delta$, consisting of formulas given by the grammar:

$$\phi, \psi, \dots \coloneqq A(\overline{x}) \mid x > y \mid x \geq y \mid \phi \to \psi \mid \forall x\, \psi.$$

Here A is a judgment of $\mathcal{C}$, treated as a relation symbol of arity $\mathsf{ob}(A)$.

$\mathcal{C}_{\mathsf{ind}}$ contains the structural rules (including cut), the normal intuitionistic rules for $\to$ and $\forall$, and:

$$\frac{}{\Gamma \vdash x \geq x}\geq_{\mathsf{refl}}, \quad \frac{\Gamma \vdash x \geq y \quad \Gamma \vdash y \geq z}{\Gamma \vdash x \geq z}\geq_{\mathsf{trans}}, \quad \frac{\Gamma \vdash x > y}{\Gamma \vdash x \geq y}\geq_{\mathsf{subsum}},$$

$$\frac{\Gamma \vdash x \geq y \quad \Gamma \vdash y > z}{\Gamma \vdash x > z}>_{\mathsf{extend}_0}, \quad \frac{\Gamma \vdash x > y \quad \Gamma \vdash y \geq z}{\Gamma \vdash x > z}>_{\mathsf{extend}_1},$$

$$x' \text{ fresh } \frac{\Gamma, \forall x' \, (x > x' \to \phi[x'/x]) \vdash \phi}{\Gamma \vdash \forall x\, \phi}>_{\mathsf{ind}_x}.$$

Inductive proof system

Lastly, for every $\mathcal{C}$-rule

$$\frac{A_i \qquad \text{for every } i < n}{A}\, r,$$

with size-change graphs $(G_i : A \to A_i)_{i<n}$, we have a $\mathcal{C}_{\mathsf{ind}}$-rule

$$\overline{y} \text{ fresh } \frac{\Gamma, \overline{x} \gtrsim_{G_i} \overline{y} \vdash A_i(\overline{y}) \qquad \text{for every } i < n}{\Gamma \vdash A(\overline{x})}\, r.$$

Here $\overline{x} \gtrsim_{G_i} \overline{y}$ contains (strict/non-strict) inequalities according to G_i.

Induction on the entire sequent

Instead of

$$x' \text{ fresh } \frac{\Gamma, \forall x' \, (x > x' \to \phi[x'/x]) \vdash \phi}{\Gamma \vdash \forall x \, \phi} >_{\mathsf{ind}_x}$$

we will use the following variation

$$x', \overline{z}' \text{ fresh } \frac{\Gamma, \forall x', \overline{z}' \, (x > x' \to (\Gamma \to \delta)[x', \overline{z}'/x, \overline{z}]) \vdash \delta}{\Gamma \vdash \delta} >_{\mathsf{ind}'_x}$$

These two induction rules are interderivable; one can derive $>_{\mathsf{ind}'_x}$ by applying $>_{\mathsf{ind}_x}$ to the formula representing the entire sequent:

$$\phi := \forall \overline{z} \, (\Gamma \to \delta).$$

General Idea

We will transform a (possibly infinite) $\mathcal{C}$-proof π of A_0 into a finite $\mathcal{C}_{\mathsf{ind}}$-proof π_{ind} of $\vdash A_0(\overline{x}_0)$:

- We first translate π into a (possibly infinite) derivation $\pi^{\mathsf{inf}}_{\mathsf{ind}}$ in $\mathcal{C}_{\mathsf{ind}}$:

$$\dfrac{\dfrac{\dfrac{\vdots}{A_3}}{A_2}\,r_2}{A_1}\,r_1}{A_0}\,r_0 \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\vdots}{\overline{x_0} \gtrsim_{G_0} \overline{x_1}, \overline{x_1} \gtrsim_{G_1} \overline{x_2}, \overline{x_2} \gtrsim_{G_2} \overline{x_3} \vdash A_3(\overline{x}_3)}\,r_2}{\overline{x_0} \gtrsim_{G_0} \overline{x_1}, \overline{x_1} \gtrsim_{G_1} \overline{x_2} \vdash A_2(\overline{x}_2)}\,r_1}{\overline{x_0} \gtrsim_{G_0} \overline{x_1} \vdash A_1(\overline{x_1})}\,r_0}{\vdash A_0(\overline{x_0})}$$

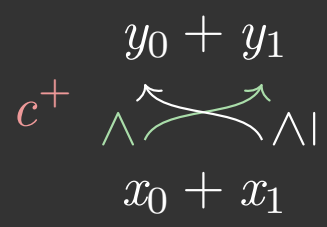
- We will 'cut short' the infinite branches of $\pi^{\mathsf{inf}}_{\mathsf{ind}}$ by introducing induction hypotheses below, and applying them above, yielding π_{ind}.

  For this, we use a nice cyclic representation π_{cyc} of π: we introduce an induction hypothesis at the sprout and apply it at the bud.

Importance of progress

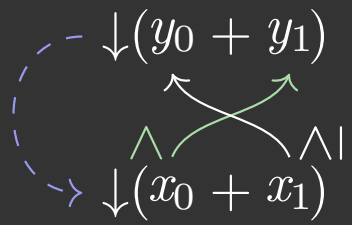
Consider the swapped addition function:

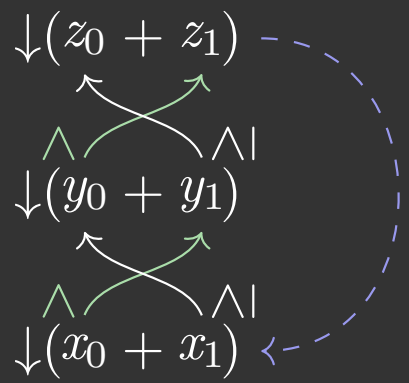
$$0 + x_1 \coloneqq x_1,$$
$$\mathsf{suc}(x_0') + x_1 \coloneqq \mathsf{suc}(\underbrace{x_1 + x_0'}_{c^+}),$$



$$\dfrac{\dfrac{\vdots}{\downarrow +}}{\downarrow +} +.$$

The proof π has (among others) the following two cyclic representations, where we have visualized the size-change graphs:





The second representation can be used to prune the infinite derivation $\pi_{\mathsf{ind}}^{\mathsf{inf}}$ into a finite proof π_{ind} (since there is an input with progress).

Importance of progress

The infinite derivation $\pi_{\mathsf{ind}}^{\mathsf{inf}}$ is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{x_0 > y_1, x_1 \geq y_0, y_0 > z_1, y_1 \geq z_0 \vdash \downarrow(z_0 + z_1)}+}{x_0 > y_1, x_1 \geq y_0 \vdash \downarrow(y_0 + y_1)}+}{\vdash \downarrow(x_0 + x_1)}+$$

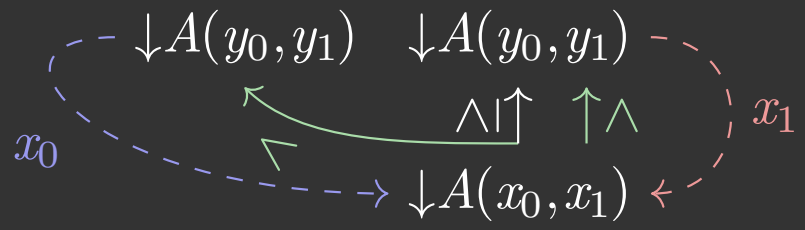
The cyclic representation of the previous slide yields π_{ind}:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\text{use } \mathsf{hyp}(x_0) \text{ on } x_0' := z_0 \text{ and } x_1' := z_1)}{\mathsf{hyp}(x_0), x_0 > y_1, x_1 \geq y_0, y_0 > z_1, y_1 \geq z_0 \vdash \downarrow(z_0 + z_1)}+}{\mathsf{hyp}(x_0), x_0 > y_1, x_1 \geq y_0 \vdash \downarrow(y_0 + y_1)}+}{\mathsf{hyp}(x_0) \vdash \downarrow(x_0 + x_1)} >_{\mathsf{ind}'_{x_0}},$$

where $\mathsf{hyp}(x_0) := \forall x_0' x_1' \, (x_0 > x_0' \to \downarrow(x_0' + x_1'))$.

Importance of induction order

The termination proof of the Ackermann function has cyclic repr:



We introduce an induction hypothesis for x_0 and x_1 in order:

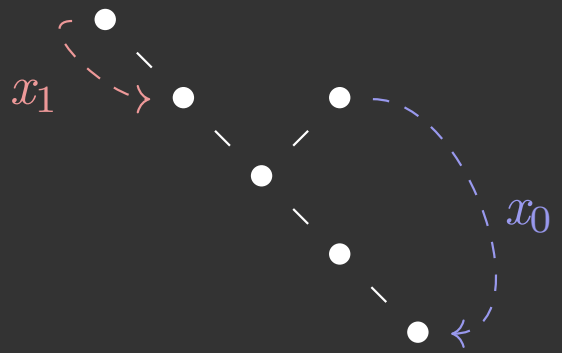
$$\dfrac{\dfrac{\dfrac{\dfrac{(\text{use } \mathsf{hyp}_0(x_0) \text{ on } x_i' := y_i)}{\ldots, x_0 > y_0 \vdash \downarrow A(y_0, y_1)} \quad \dfrac{(\text{use } \mathsf{hyp}_1(x_1) \text{ on } x_i' := y_i)}{\ldots, x_0 \geq y_0, x_1 > y_1 \vdash \downarrow A(y_0, y_1)}}{\mathsf{hyp}_0(x_0), \mathsf{hyp}_1(x_1) \vdash \downarrow A(x_0, x_1)}\, A}{\mathsf{hyp}_0(x_0) \vdash \downarrow A(x_0, x_1)}\, >_{\mathsf{ind}'_{x_1}}}{\vdash \downarrow A(x_0, x_1)}\, >_{\mathsf{ind}'_{x_0}},$$

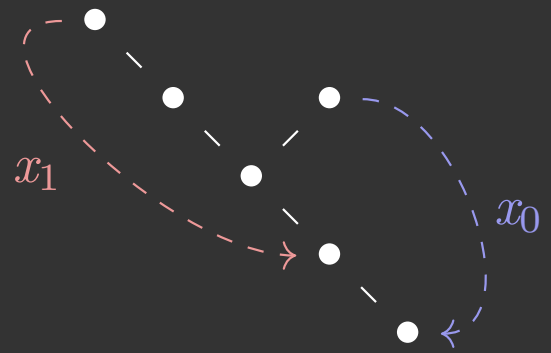
where the induction hypotheses are:

$$\mathsf{hyp}_0(x_0) := \forall x_0' \, \forall x_1' \, (x_0 > x_0' \to \downarrow A(x_0', x_1')),$$
$$\mathsf{hyp}_1(x_1) := \forall x_0' \, \forall x_1' \, (x_1 > x_1' \to \mathsf{hyp}_0(x_0') \to \downarrow A(x_0', x_1')).$$

Importance of forgetting

Consider the following two forms of cyclic representations:





For both, if we want to transform them into finite $\mathcal{C}_{\text{ind}}$-proofs, we have to introduce an induction hypothesis at the root for x_0:

- in the first example, we can forget the induction hypothesis for x_0 (using weakening) before introducing the induction hypothesis for x_1;
- in the second example this is not possible, so in this example the back-edge for x_1 has to 'preserve' x_0 in some way.

Non-linear

The main technical difficulty is finding a suitable cyclic representation:

- We use annotations based on 'reset proofs' and the 'Safra construction' to find a cyclic representation with an induction order.
- We use a technique of Sprenger and Dam to unfold the tree until the order of sprouts in the tree respects the induction order.

Here we modify reset proofs in one important way:

- Instead doing a reset whenever we have a 'cover', we wait until we have a 'uniform cover'.

This allows us to remove the restriction that the well-founded relation $\leq$ is a linear order: we only assume that it is a quasi-order.

Conclusion

Theorem

For every $\mathcal{C}$-proof π of A, there exists a $\mathcal{C}_{\mathsf{ind}}$-proof π_{ind} of $\vdash A(\overline{x}_0)$. Moreover, π_{ind} preserves the structure of π: the $\mathcal{C}$-rules of π_{ind} form a finite subtree of π.

We also have a multi-sorted version of this theorem: here the trace objects of $\mathcal{C}$ are given sorts, and $\mathcal{C}_{\mathsf{ind}}$ is a multi-sorted first-order logic.

Future Work

- Extend to a mix between inductive and coinductive sorts:
  - Either by allowing both inductive and coinductive traces.
  - Or through ordinal approximations or sized types.
- See whether the results hold in a constructive metatheory.
- Apply our result to dependent type theory: show that size-change termination is conservative over primitive elimination rules (extending the results in the PhD theses of McBride and Cockx):
  - Here we have some ideas for inductive types.
  - We want to extend to cover indexed inductive types.

Preprint

https://arxiv.org/abs/2602.12054

Application: cyclic Heyting/Peano arithmetic

Our results can be used to transform a CHA (cyclic Heyting arithmetic) proof into a normal HA (Heyting arithmetic) proof:

- First we translate the CHA-proof into a $\mathsf{CHA}_{\mathsf{ind}}$-proof.
- Because HA and CHA are already first-order theories containing intuitionistic rules for $\to$ and $\forall$, and we can define

$$(n \leq m) \coloneqq \exists x\,(n + x = m), \qquad (n < m) \coloneqq (\mathsf{suc}(n) \leq m),$$

  we can translate the $\mathsf{CHA}_{\mathsf{ind}}$-proof into a HA-proof.

The same strategy gives a transformation from CPA into PA.