

Least and greatest fixed points: proof theory and applications

David Baelde

University of Minnesota

Sept 18, 2010

What is proof theory?

Studying logics through their proofs

Constraints

Sequent calculus, cut-elimination, focusing. . .

Rewards

- ▶ Modular: many logics
- ▶ General:
 logic programming, model-checking, theorem proving
- ▶ Composable certificates

Arithmetic

Natural numbers

$0, 1, 2, \dots$

Induction

Pn holds for any natural number n if:

- ▶ $P0$ holds.
- ▶ $P(sm)$ holds whenever Pm does.

Arithmetic

Natural numbers

Built by iterating the following transformation on \emptyset :

$$N \mapsto \{0\} \cup \{sx : x \in N\}$$

Induction

Pn holds for any natural number n if

- ▶ Px holds under the assumption that it holds for the predecessors of x in the iteration.

Fixed points everywhere

Least fixed points: inductive definitions

- ▶ Natural numbers.
- ▶ Lists, trees, programs.
- ▶ Computation.

Greatest fixed points: coinductive definitions

- ▶ Infinite data structures, *e.g.*, streams.
- ▶ Behavioral equivalences.

Universal reasoning principles: induction and coinduction.

The proof-theory of fixed points

Example: Natural numbers

$$\begin{aligned} B_{nat} N x &:= x = 0 \vee (\exists y. x = sy \wedge Ny) \\ nat &:= \mu B_{nat} \end{aligned}$$

The rules of μ LJ

$$\frac{\Gamma, B(\mu B)\vec{t} \vdash P}{\Gamma, \mu B\vec{t} \vdash P}$$

$$\frac{\Gamma \vdash B(\mu B)\vec{t}}{\Gamma \vdash \mu B\vec{t}}$$

$$\frac{\Gamma[x/0] \vdash P[x/0] \quad \Gamma[sy/x], nat y \vdash P[sy/x]}{\Gamma, nat x \vdash P}$$

$$\frac{}{\Gamma \vdash nat 0} \quad \frac{\Gamma \vdash nat x}{\Gamma \vdash nat (sx)}$$

The proof-theory of fixed points

Example: Natural numbers

$$\begin{aligned} B_{nat} N x &:= x = 0 \vee (\exists y. x = sy \wedge Ny) \\ nat &:= \mu B_{nat} \end{aligned}$$

The rules of μ LJ

$$\frac{BS\vec{x} \vdash S\vec{x}}{\mu B\vec{t} \vdash S\vec{t}} \qquad \frac{\Gamma \vdash B(\mu B)\vec{t}}{\Gamma \vdash \mu B\vec{t}}$$
$$\frac{\vdash S0 \quad S y \vdash S(s y)}{nat t \vdash St}$$

The proof-theory of fixed points

Example: Natural numbers

$$\begin{aligned} B_{nat} N x &:= x = 0 \vee (\exists y. x = sy \wedge Ny) \\ nat &:= \mu B_{nat} \end{aligned}$$

The rules of μ LJ

$$\frac{BS\vec{x} \vdash S\vec{x}}{\mu B\vec{t} \vdash S\vec{t}} \quad \frac{\Gamma \vdash B(\mu B)\vec{t}}{\Gamma \vdash \mu B\vec{t}}$$
$$\frac{\Gamma, B(\nu B)\vec{t} \vdash P}{\Gamma, \nu B\vec{t} \vdash P} \quad \frac{S\vec{x} \vdash BS\vec{x}}{S\vec{t} \vdash \nu B\vec{t}}$$

Overview

- ▶ A very rich logic, undecidable
- ▶ Cut-elimination: consistency, composability
- ▶ Focused proofs: meaningful skeletons
- ▶ Automated reasoning:
 - ▶ logic programming (Prolog)
 - ▶ model-checking (Bedwyr)
 - ▶ inductive theorem proving (Tac)

A quick example

Definition (Similarity)

The greatest relation \mathcal{R} such that whenever $p\mathcal{R}q$ and $p \xrightarrow{\alpha} p'$, there is a q' such that $q \xrightarrow{\alpha} q'$ and $p'\mathcal{R}q'$.

In Bedwyr

```
coinductive sim p q :=  
  pi a \ p' \ step p a p' =>  
    sigma q' \ step q a q', sim p' q'.
```

- ▶ Extends to π -calculus (bindings in transition labels)
- ▶ Bisimulation checker ((in)finite case)

In Tac

Automatically prove that simulation is transitive, that bisimulation is reflexive, etc.

An illustration

Regular formulas

- ▶ It's not about pushing **symbols**, but finding **structures**
- ▶ A research program: **verification in proof theory**

Verification

Fixed points are everywhere. For example. . .

Model-checking

- ▶ Does a system satisfy a specification?
- ▶ $M \models S$
- ▶ Often translated to automata inclusion $[M] \subseteq [S]$

How do you **prove** an inclusion?

$$[M]_X \vdash [S]_X$$

What is the structure of inclusion?

NFA: Definitions

Non-deterministic finite automata

- ▶ Alphabet $\Sigma = \{\alpha, \beta, \gamma, \dots\}$
- ▶ Finite set of states
- ▶ Distinguished **initial** and **final** states
- ▶ Transition relation $s \rightarrow^\alpha q$

Definition

If Q is a set of states,

$Q \rightarrow^\alpha Q'$ iff each state of Q' is reachable from Q .

In other words, $Q' \subseteq \alpha^{-1}Q$.

Structure of inclusion

Definition (Multi-simulation)

A multi-simulation between two automata (A, T, I, F) and (B, T', I', F') is a relation $\mathfrak{R} \subseteq A \times \wp(B)$ such that whenever $p \mathfrak{R} Q$:

- ▶ if p is final, then there must be a final state in Q ;
- ▶ for any α and p' such that $p \rightarrow^\alpha p'$ there exists Q' such that $Q \rightarrow^\alpha Q'$ and $p' \mathfrak{R} Q'$.

Multi-simulations are post-fixed points.

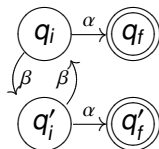
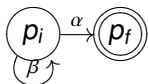
There is a greatest one: call it **multi-similarity**.

Proposition (Multi-similarity is inclusion)

$\mathcal{L}(p) \subseteq \mathcal{L}(Q)$ if and only if $p \mathfrak{R} Q$ for some multi-simulation \mathfrak{R} .

Examples

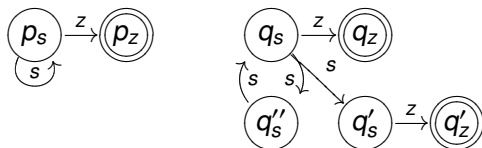
Consider the following two automata:



States p_i et q_i accept the same language. Proof:

$$\mathfrak{R} = \{(p_i, \{q_i\}), (p_i, \{q'_i\}), (p_f, \{q_f\}), (p_f, \{q'_f\})\}$$

Examples



We prove $\mathcal{L}(p_s) \subseteq \mathcal{L}(q_s)$:

$$\mathfrak{R} = \{(p_s, \{q_s\}), (p_s, \{q'_s, q''_s\}), (p_z, \{q_z\}), (p_z, \{q'_z\})\}$$

(Did we prove $\forall x. \text{nat } x \supset \exists h. \text{half } x \ h$?!)

Conclusion

Summary

- ▶ Design a good framework
- ▶ Import techniques, understand them
- ▶ Generalize, interoperate

Future work

- ▶ Extend: Büchi, tree and alternating automata
- ▶ More automated reasoning, loop detection in Bedwyr