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, …

Induction

$P_n$ holds for any natural number $n$ if:

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

\[ B_{nat} N x := x = 0 \lor (\exists y. x = sy \land Ny) \]
\[ nat := \mu B_{nat} \]

The rules of \( \mu \text{LJ} \)

\[
\frac{\Gamma, B(\mu B)\overrightarrow{t} \vdash P}{\Gamma, \mu B\overrightarrow{t} \vdash P}
\]
\[
\frac{\Gamma \vdash B(\mu B)\overrightarrow{t}}{\Gamma \vdash \mu B\overrightarrow{t}}
\]
\[
\frac{\Gamma[x/0] \vdash P[x/0] \quad \Gamma[sy/x], nat y \vdash P[sy/x]}{\Gamma, nat x \vdash P}
\]
\[
\frac{\Gamma \vdash nat 0}{\Gamma \vdash nat (sx)}
\]
The proof-theory of fixed points

Example: Natural numbers

\[ \begin{align*}
B_{nat} \; N \; x & \; := \; x = 0 \lor (\exists y. \; x = sy \land Ny) \\
nat & \; := \; \mu B_{nat}
\end{align*} \]

The rules of \( \mu \text{LJ} \)

\[ \begin{align*}
BS \bar{x} & \vdash S \bar{x} \\
\mu B \bar{t} & \vdash S \bar{t} \\
\Gamma & \vdash B(\mu B) \bar{t} \\
\Gamma & \vdash \mu B \bar{t}
\end{align*} \]

\[ \vdash S0 \quad S \; y & \vdash S \; (s \; y) \\
nat \; t & \vdash St
\]
The proof-theory of fixed points

Example: Natural numbers

\[ B_{nat} \ N \ x \ := \ x = 0 \lor (\exists y. x = sy \land Ny) \]
\[ nat \ := \ \mu B_{nat} \]

The rules of \( \mu \text{LJ} \)

\[
\begin{align*}
BSx \vdash Sx & \quad \mu Bt \vdash St \\
\mu Bt \vdash St & \quad \Gamma \vdash B(\mu B)t \\
\Gamma, B(\nu B)t \vdash P & \quad Sx \vdash BSx \\
Sx \vdash BSx & \quad \Gamma, \nu Bt \vdash P \\
\end{align*}
\]
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 $R$ such that whenever $pRq$ and $p \xrightarrow{\alpha} p'$, there is a $q'$ such that $q \xrightarrow{\alpha} q'$ and $p'Rq'$.

In Bedwyr

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

▶ Extends to $\pi$-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?
Non-deterministic finite automata

- Alphabet \( \Sigma = \{ \alpha, \beta, \gamma, \ldots \} \)
- 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 \(\mathcal{R} \subseteq A \times \mathcal{P}(B)\) such that whenever \(p \mathcal{R} Q\):

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

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 \(p \mathcal{R} Q\) for some multi-simulation \(\mathcal{R}\).
Examples

Consider the following two automata:

States $p_i$ et $q_i$ accept the same language. Proof:

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

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

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

(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