Towards a Unified Proof Framework for Automated Fixpoint Reasoning using Matching Logic

Xiaohong Chen, Minh-Thai Trinh, Nishant Rodrigues, Lucas Pena, Grigore Rosu

University of Illinois at Urbana-Champaign, USA
Advanced Digital Sciences Center, Illinois at Singapore, Singapore

OOPSLA 2020
Fixpoints are Everywhere...

- Heaps: $ll(x, y) \land (y = nil) \rightarrow list(x)$
- Streams: $zip(zeros, ones) = blink = 01010101 \ldots$
- Terms: $plus(m, n) = plus(n, m)$ on term-algebra $T_{zero, succ}$
- Temporal properties: $\varphi \land (\varphi \rightarrow □ \varphi) \rightarrow □\varphi$
- Program correctness: $\varphi_{pre} \Rightarrow \varphi_{post}$
- ... ...
- However, there is no unified proof framework aimed at automated fixpoint reasoning in all the above domains.
A Unified Proof Framework for Automated Reasoning

We use matching logic as the base logic.
Why Matching Logic?

- Matching Logic: The Underlying Core Logic
  - First-Order Logic with Least Fixpoints
  - Separation Logic with Recursive Definitions
  - Reachability Logic
  - Modal μ-Logic
  - Temporal Logics (LTL, CTL, CTL*, ...)
  - Polyadic and/or Hybrid Modal Logic
  - Dynamic Logic
  - Hoare Logic
  - Normal Modal Logic
  - Propositional Logic
  - Initial Algebra Semantics
  - Order-Sorted Algebras
  - Many-Sorted Algebras
  - Equational Logic
  - λ-Calculus

Matching Logic Prover

- Matching Logic Theory of FOL-LFP
  - FOL-LFP Prover

- Matching Logic Theory of Separation Logic with Fixpoints
  - Separation Logic Prover

- Matching Logic Theory of Reachability Logic
  - Reachability Logic Prover (Program Verifier)

- Matching Logic Theory of LTL/CTL
  - LTL/CTL Prover
Matching Logic in a Nutshell

- Examples of matching logic patterns (in various logical theories):
  - `cons(x, nil)`
  - `x \mapsto y * list(y)`
  - `\exists y. x \mapsto y * list(y)`
  - `\Box \varphi \rightarrow \Diamond \varphi`, where `\Box \varphi \equiv \mu X. (\varphi \land \Diamond X)`.
  - `\langle \langle \text{while}(n \geq 0) \ldots \rangle_{\text{code}} \langle n \mapsto 100, sum \mapsto 0 \rangle_{\text{state}} \rangle_{\text{cfg}}`
  - `\varphi_{\text{pre}} \Rightarrow \varphi_{\text{post}}`

- The matching logic theory $\Gamma^{SL}$ for separation logic.
- The matching logic theory $\Gamma^{LTL}$ for linear temporal logic (LTL).
- The matching logic theory $\Gamma^{RL}$ for reachability logic (program verification).
Existing Matching Logic Proof System

- The existing proof system is **not suitable** for proof automation.
  - It gives **too much freedom** in proof search.
- Two proof rules in the existing proof system:

  ![Modus Ponens and Knaster-Tarski Rules]

  - We need to “guess” the premise $\varphi_1$

- It requires LHS to be a fixpoint, but in practice, the LHS often has the form $C[\mu X. \varphi] \rightarrow \psi$, where the fixpoint occurs within a context. E.g.,:

  $\text{listseg}(x, y) * \text{list}(y) \rightarrow \text{list}(x)$

- We need a new set of proof rules with fewer branching rules and knows how to deal with contexts.
Our New Proof Framework

(a) Proof Rules for ML Fixpoint Reasoning

(b) Breakdown of Rule (κτ) in Fig. 2a

Fig. 2. Automatic Proof Framework for ML Fixpoint Reasoning (where $p(\bar{x}) \equiv_{ip} \bigvee_i \phi_i$)
Key Rule: LFP (Park Induction)

Recursive definition: 
\[
\begin{align*}
\rho(\vec{x}) &= \text{LFP } \exists \vec{x}_1. \phi_1(\vec{x}_1) \land \cdots \land \exists \vec{x}_m. \phi_m(\vec{x}_1, \vec{x}_m) \\
\text{(LFP)} \quad \exists \vec{x}_1. \phi_1[\psi/p] &\implies \psi \quad \cdots \quad \exists \vec{x}_m. \phi_m[\psi/p] &\implies \psi \\
\rho(\vec{x}) &\implies \psi
\end{align*}
\]

• This rule lies at the core of many inductive proof systems.
• It allows us to prove, e.g., \( ll(x, y) \rightarrow lr(x, y) \).
• **Limitation**: LHS must be a fixpoint.
• How to prove these?
  • \( ll(x, y) \ast \text{list}(x) \rightarrow \text{list}(y) \)
  • \( \langle \text{while}(n \geq 0) \rangle_{\text{code}} \langle n \mapsto N, \text{sum} \mapsto 0 \rangle_{\text{state}} \) 

Note that fixpoints occur within a context.
Reasoning Fixpoints within Contexts

• Proof Goal: \( ll(x, y) \ast list(y) \rightarrow list(x) \) consists of
  • A fixpoint \( ll(x, y) \)
  • A context \( \square \square \equiv \square \ast list(y) \)

• We (WRAP) the context and move it to the RHS:
  • \( ll(x, y) \rightarrow \exists h: Heap. (h \land (h \ast list(y) \rightarrow list(x))) \)

The set of all heaps \( h \) such that \( h \ast list(y) \rightarrow list(x) \).

• We call the above RHS a contextual implication, abbreviated:
  • \( ll(x, y) \rightarrow (C \rightarrow list(x)) \)

• Now, LHS is a fixpoint and we can apply (LFP) in the usual way.
Contextual Implications

• Let $C[\square]$ be a context and $\psi$ be any pattern.

• **Contextual implication** $C \leadsto \psi \equiv \exists x. \ x \land (C[x] \rightarrow \psi)$
  • A matching logic pattern
  • Matched by all $x$ such that $\psi$ holds if within context $C$

• Key property (wrapping and unwrapping):

\[
\begin{align*}
\vdash & \ C[\varphi] \rightarrow \psi & \text{(WRAP) context } C \\
\vdash & \varphi \rightarrow (C \leadsto \psi) & \text{(UNWRAP) context } C
\end{align*}
\]

• $C$ can be any context (some mild conditions should be satisfied).

• Separating implication is a special case.
  • Let $C_\varphi \equiv \square \ast \varphi$, then separating implication $\varphi \ast \psi = C_\varphi \rightarrow \psi$
  • Separation logic rule (ADJ) is a special case of (WRAP)/(UNWRAP).
  • (ADJ). $\vdash h \ast \varphi \rightarrow \psi$ iff $\vdash h \rightarrow (\varphi \ast \psi)$
Evaluation

• We consider four representative logical systems for fixpoint reasoning:
  • Separation logic (with recursive predicates);
    • Proved $265/280$ benchmark tests in SL-COMP’19.
  • Linear temporal logic (LTL);
    • Proved the axioms in the complete LTL proof system.
  • FOL with least fixpoints (LFP) and reachability logic (for program verification)
    • Proved the correctness of the SUM program (computing the sum from $1$ to $n$).
• Main bottlenecks (future research):
  • Improve non-fixpoint reasoning;
  • Design smarter proof strategies/heuristics.
Conclusion
A Unified Proof Framework for Automated Reasoning based on Matching Logic