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

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

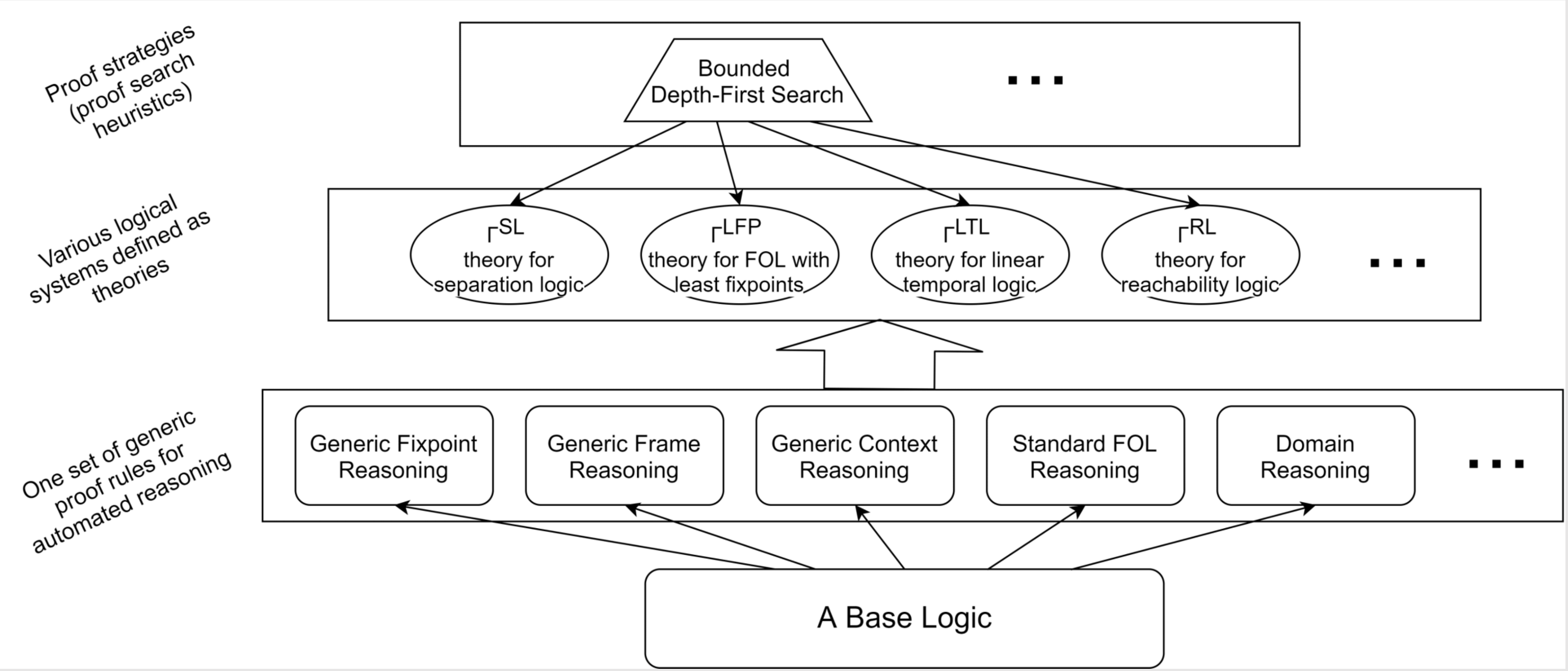
University of Illinois at Urbana-Champaign, USA
Advanced Digital Sciences Center, Illinois at Singapore, Singapore

OOPSLA 2020

Fixpoints are Everywhere…

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

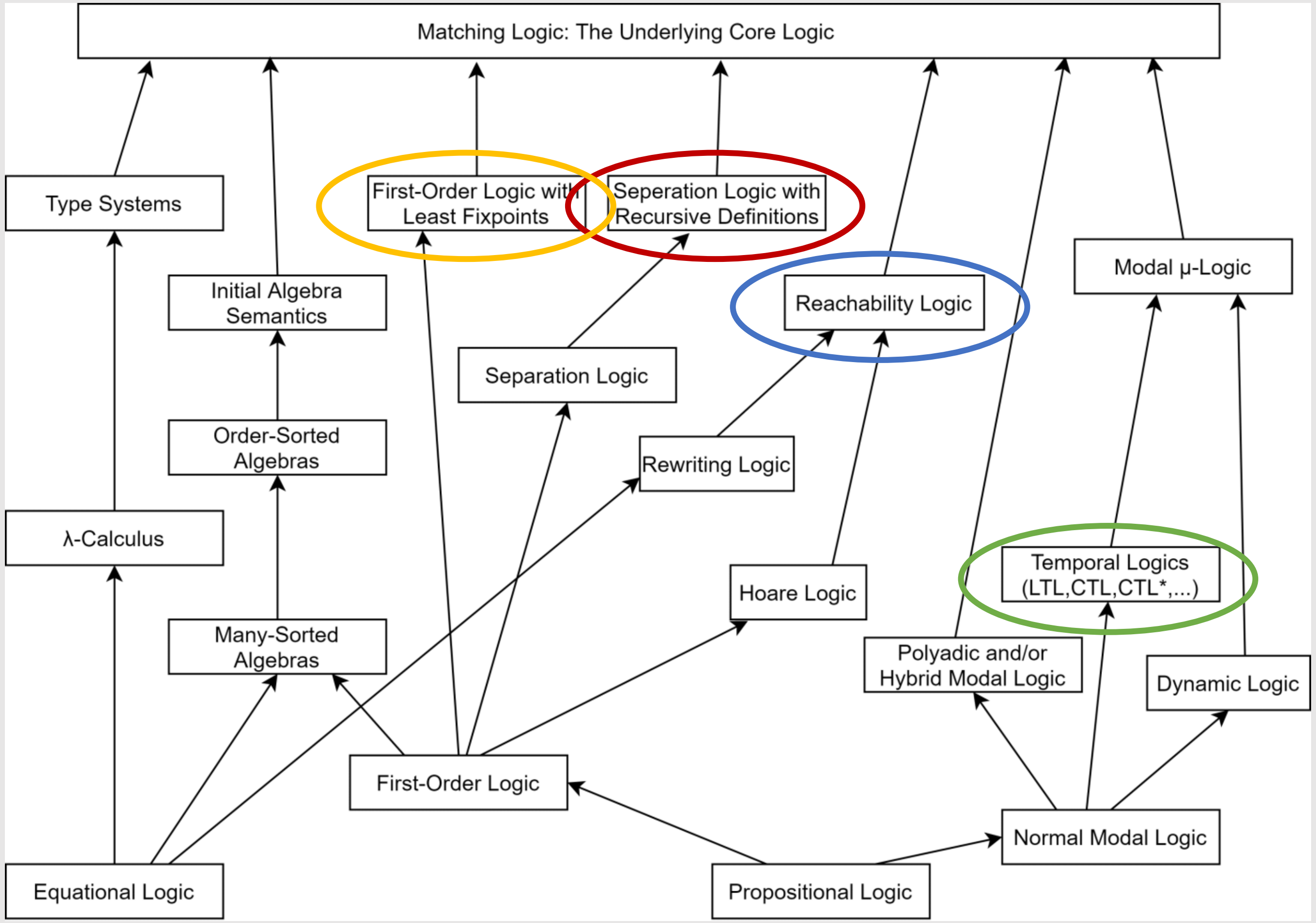
A Unified Proof Framework for Automated Reasoning

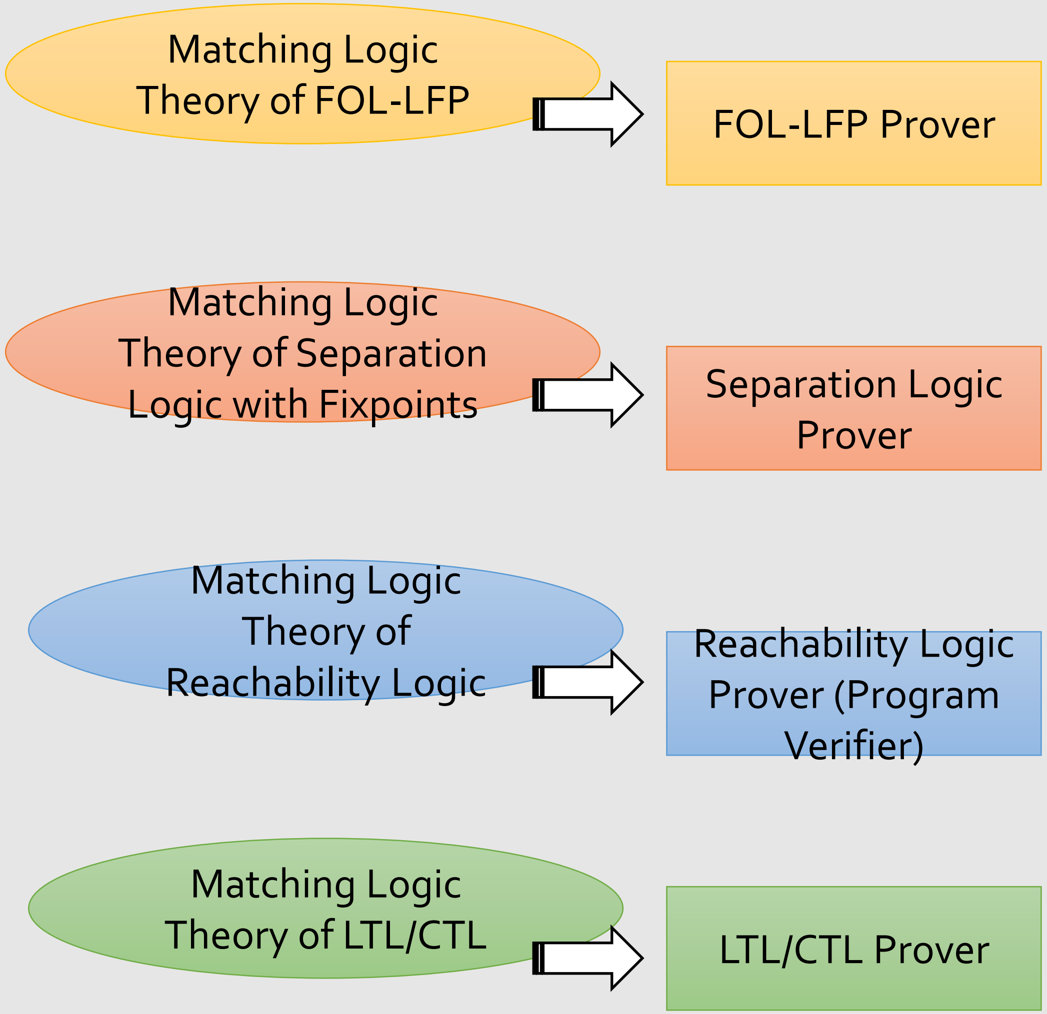


We use **matching logic** as the base logic.

Why Matching Logic?

Matching Logic Prover





Matching Logic in a Nutshell

$$\text{Matching Logic} = \boxed{\text{A unified syntax of } \textit{patterns}} + \boxed{\text{A unified semantics based on } \textit{pattern matching}} + \boxed{\text{One fixed Hilbert } \textit{proof system}}$$

$$\varphi ::= \underbrace{x \mid \sigma(\varphi_1, \dots, \varphi_n)}_{\text{structures}} \mid \underbrace{\varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \mid \neg\varphi \mid \varphi_1 = \varphi_2 \mid \varphi_1 \subseteq \varphi_2}_{\text{logical constraints}} \mid \underbrace{\exists x.\, \varphi \mid \forall x.\, \varphi}_{\text{quantification}} \mid \underbrace{X \mid \mu X.\, \varphi \mid \nu X.\, \varphi}_{\text{fixpoints}}$$

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

the matching logic theory Γ^{SL} for separation logic.

the matching logic theory Γ^{LTL} for linear temporal logic (LTL)

the matching logic theory Γ^{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:

$$\text{(Modus Ponens)} \quad \frac{\varphi_1 \quad \varphi_1 \to \varphi_2}{\varphi_2}$$

$$\text{(Knaster-Tarski)} \quad \frac{\varphi[\psi/X] \to \psi}{\mu X\,.\,\varphi \to \psi}$$

We need to "guess" the premise φ_1

It requires LHS to be a fixpoint, but in practice, the LHS often has the form $C[\mu X.\varphi] \to \psi$, where the fixpoint occurs within a context. E.g.,: $listseg(x,y) * list(y) \to list(x)$

- We need a new set of proof rules with fewer branching rules and knows **how to deal with contexts**.

Our New Proof Framework

$$(\textsc{elim-}\exists)\quad \frac{\varphi \to \psi}{(\exists x.\, \varphi) \to \psi} \quad \text{if } x \notin \mathrm{FV}(\psi)$$

$$(\textsc{smt})\quad \frac{\mathsf{True}}{\varphi \to \psi} \quad \text{if } \vDash_{\mathsf{SMT}} \varphi \to \psi$$

$$(\textsc{pm})\quad \frac{\varphi \to \psi\theta}{\varphi \to \exists \tilde{y}.\, \psi} \quad \text{where } \theta \in \mathsf{pm}(\varphi, \psi, \tilde{y}) \text{ matches } \varphi \text{ with } \psi$$

$$(\textsc{match-ctx})\quad \frac{C_{rest}[\varphi'\theta] \to \psi}{C_o[\forall \tilde{y}.\, (C' \multimap \varphi')] \to \psi} \quad \text{where } (C_{rest}, \theta) = \mathsf{cm}(C_o, C', \tilde{y})$$

$$(\textsc{frame})\quad \frac{\varphi \to \psi}{C[\varphi] \to C[\psi]}$$

$$(\textsc{unfold-r})\quad \frac{\varphi \to C[\varphi_i]}{\varphi \to C[p(\tilde{x})]}$$

$$(\textsc{kt})\quad \frac{\text{Composition of Rules in Fig. 2b}}{\varphi \to \psi}$$

(a) Proof Rules for ML Fixpoint Reasoning

$$(\textsc{wrap})\quad \frac{p(\tilde{x}) \to (C \multimap \psi)}{C[p(\tilde{x})] \to \psi}$$

$$(\textsc{intro-}\forall)\quad \frac{p(\tilde{x}) \to \forall \tilde{y}.\ (C \multimap \psi)}{p(\tilde{x}) \to (C \multimap \psi)} \quad \text{where } \tilde{y} = \mathrm{FV}(\psi) \setminus \tilde{x}$$

$$(\textsc{lfp})\quad \frac{\cdots\ \varphi_i[\forall \tilde{y}.\, (C \multimap \psi)/p] \to \forall \tilde{y}.\, (C \multimap \psi)}{p(\tilde{x}) \to \forall \tilde{y}.\, (C \multimap \psi)}$$

$$(\textsc{elim-}\forall)\quad \frac{\varphi \to (C \multimap \psi)}{\varphi \to \forall y.\, (C \multimap \psi)} \quad \text{if } y \notin \mathrm{FV}(\varphi)$$

$$(\textsc{unwrap})\quad \frac{C[\varphi] \to \psi}{\varphi \to (C \multimap \psi)}$$

(b) Breakdown of Rule (KT) in Fig. 2a

Fig. 2. Automatic Proof Framework for ML Fixpoint Reasoning (where $p(\tilde{x}) =_{\mathsf{lfp}} \bigvee_i \varphi_i$)

Key Rule: LFP (Park Induction)

Recursive definition: $p(\tilde{x}) =_{\text{lfp}} \exists \tilde{x}_1 .\, \varphi_1(\tilde{x}, \tilde{x}_1) \vee \cdots \vee \exists \tilde{x}_m .\, \varphi_m(\tilde{x}, \tilde{x}_m)$

$$(\text{LFP}) \quad \frac{\exists \tilde{x}_1 .\, \varphi_1[\psi/p] \rightarrow \psi \qquad \cdots \qquad \exists \tilde{x}_m .\, \varphi_m[\psi/p] \rightarrow \psi}{p(\tilde{x}) \rightarrow \psi}$$

- 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?
  - $ll(x,y) * list(x) \rightarrow list(y)$
  - $\langle\langle \text{while}(\text{n} \geq 0) \ldots \rangle_{\text{code}} \langle n \mapsto N, sum \mapsto 0 \rangle_{\text{state}} \rangle_{\text{cfg}}$

$$\Rightarrow \left\langle \langle\, skip \,\rangle_{\text{code}} \left\langle n \mapsto 0, sum \mapsto \frac{N(N+1)}{2} \right\rangle_{\text{state}} \right\rangle_{\text{cfg}}$$

$$ll(x,y) =_{\text{lfp}} (x = y \wedge emp) \vee (x \neq y \wedge \exists t .\, x \mapsto t * ll(t,y))$$
$$lr(x,y) =_{\text{lfp}} (x = y \wedge emp) \vee (x \neq y \wedge \exists t .\, lr(x,t) * t \mapsto y)$$

Note that **fixpoints** occur within a **context**.

Reasoning Fixpoints within Contexts

- Proof Goal: $ll(x,y) * list(y) \to list(x)$ consists of
  - A **fixpoint** $ll(x,y)$
  - A **context** $C[\square] \equiv \square * list(y)$
- We (WRAP) the context and move it to the RHS:
  - $ll(x,y) \to \underbrace{\exists h{:}\, Heap.\left(h \wedge \left(h * list(y) \to list(x)\right)\right)}_{\text{The set of all heaps h such that h*list(y)->list(x).}}$
- We call the above RHS a **contextual implication**, abbreviated:
  - $ll(x,y) \to \left(C \multimap list(x)\right)$
- Now, LHS is a fixpoint and we can apply (LFP) in the usual way.

Contextual Implications

- Let $C[\square]$ be a context and ψ be any pattern.
- **Contextual implication** $C \multimap \psi \equiv \exists x.\, x \wedge (C[x] \to \psi)$
  - A matching logic pattern
  - Matched by all x such that ψ holds if within context C
- Key property (wrapping and unwrapping):

$$\vdash C[\varphi] \to \psi \qquad \underset{\text{(UNWRAP) context } C}{\overset{\text{(WRAP) context } C}{\rightleftarrows}} \qquad \vdash \varphi \to (C \multimap \psi)$$

- C can be any context (some mild conditions should be satisfied).
- Separating implication is a special case.
  - Let $C_\varphi \equiv \square * \varphi$, then separating implication $\varphi \mathrel{-\!\!*} \psi = C_\varphi \multimap \psi$
  - Separation logic rule (ADJ) is a special case of (WRAP)/(UNWRAP).
  - (ADJ). $\vdash h * \varphi \to \psi$ iff $\vdash h \to (\varphi \mathrel{-\!\!*} \psi)$

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