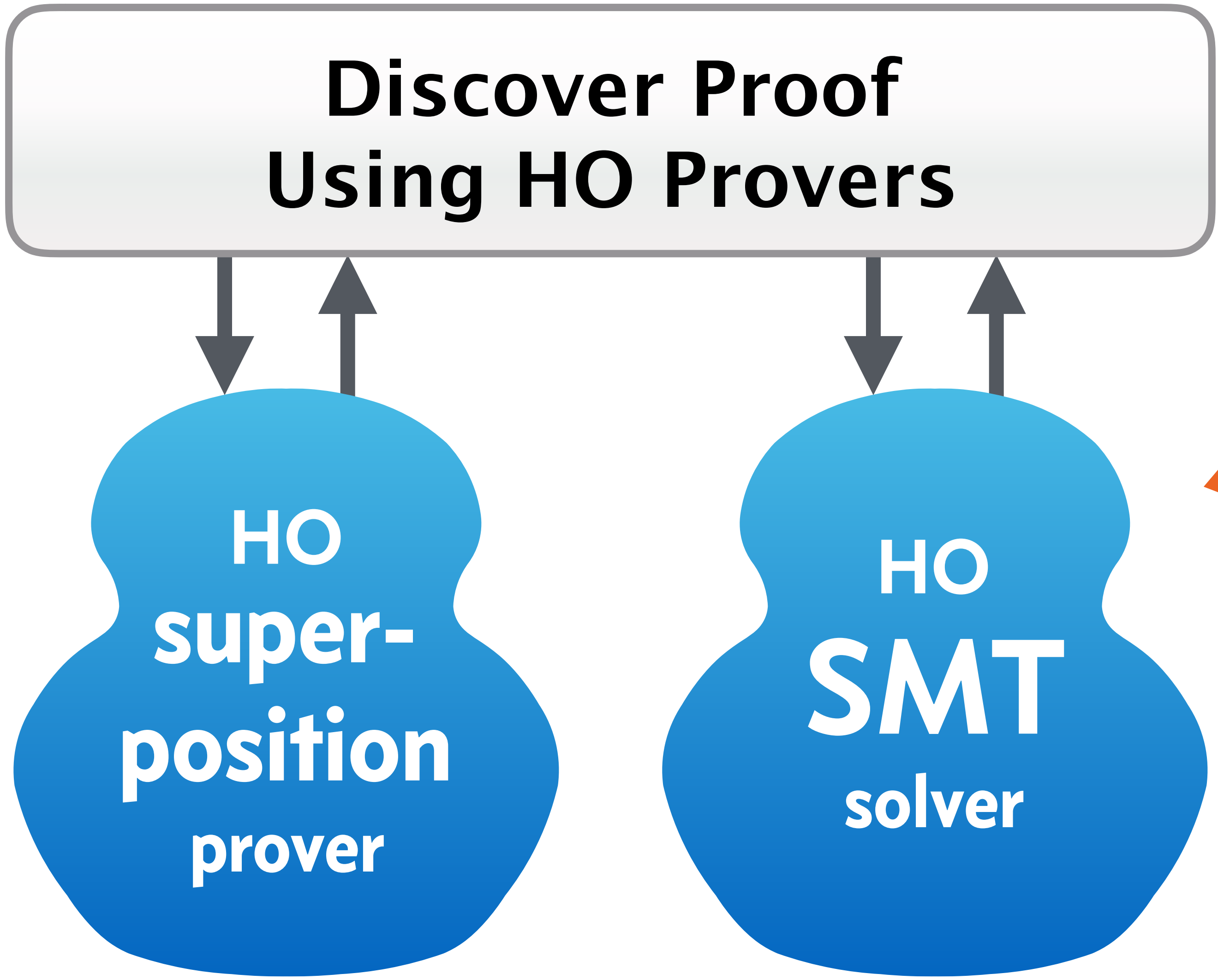
Towards
Fast Interactive Verification
through
Strong Higher-Order Automation

Jasmin Blanchette
Pascal Fontaine
Stephan Schulz
Uwe Waldmann

Vision: Take the Hard Labor out of Interactive Verification

Push button automation for proof assistants (e.g. Coq)
based on **efficient higher-order (HO) provers**



Discover Proof
Using HO Provers

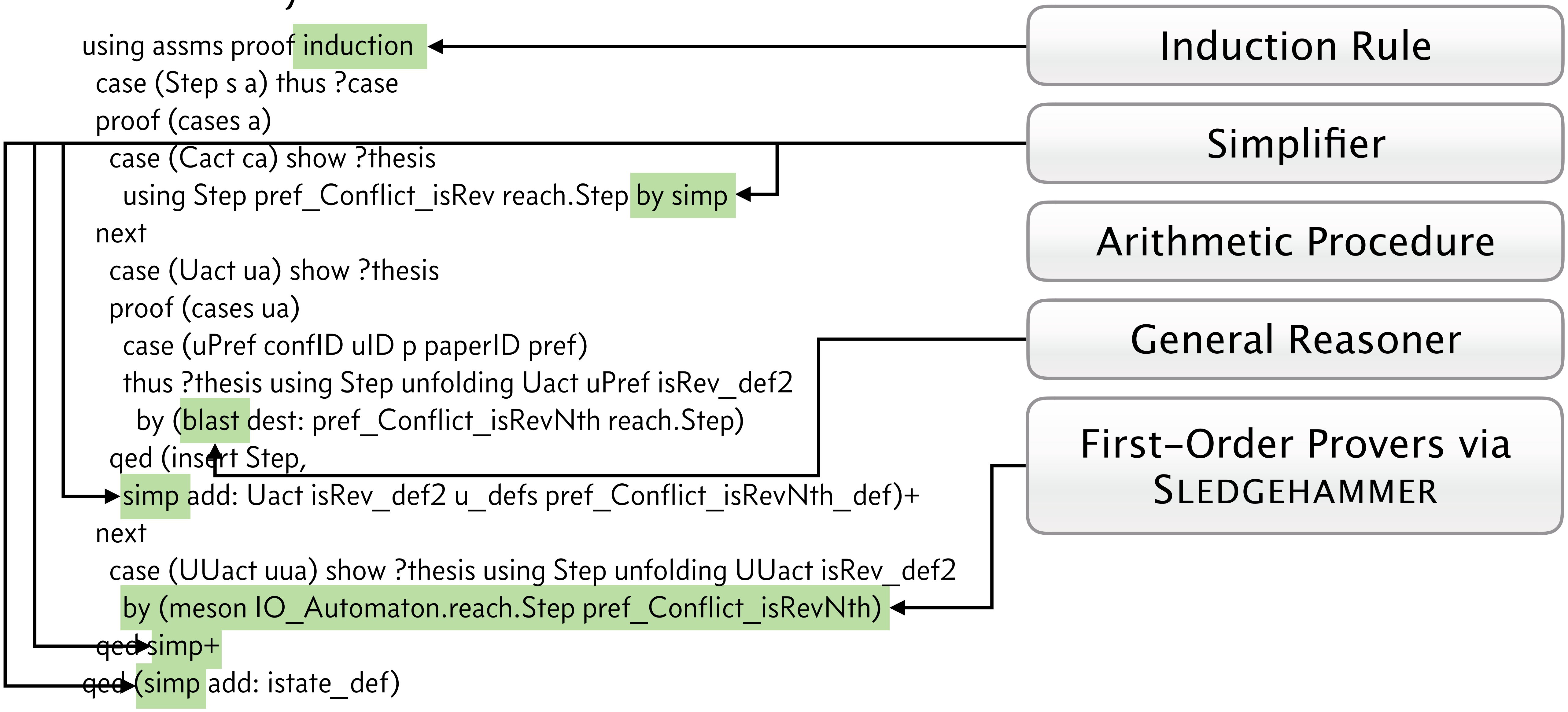
HO super-position prover

HO SMT solver

APPLICATION: A VERIFIED "EASYCHAIR"

"PC members cannot review papers if they have a conflict of interest"

Proof today:



```
using assms proof induction
 case (Step s a) thus ?case
 proof (cases a)
  case (Cact ca) show ?thesis
   using Step pref_Conflict_isRev reach.Step by simp
 next
  case (Uact ua) show ?thesis
  proof (cases ua)
   case (uPref confID uID p paperID pref)
   thus ?thesis using Step unfolding Uact uPref isRev_def2
    by (blast dest: pref_Conflict_isRevNth reach.Step)
  qed (insert Step,
   simp add: Uact isRev_def2 u_defs pref_Conflict_isRevNth_def)+
 next
  case (UUact uua) show ?thesis using Step unfolding UUact isRev_def2
   by (meson IO_Automaton.reach.Step pref_Conflict_isRevNth)
 qed simp+
qed (simp add: istate_def)
```

Induction Rule

Simplifier

Arithmetic Procedure

General Reasoner

First-Order Provers via SLEDGEHAMMER

🟩 fully automatic

Application: A Verified “EasyChair”

“PC members cannot review papers if they have a conflict of interest”

Proof today:

```
using assms proof induction
 case (Step s a) thus ?case
 proof (cases a)
  case (Cact ca) show ?thesis
   using Step pref_Conflict_isRev reach.Step by simp
 next
  case (Uact ua) show ?thesis
  proof (cases ua)
   case (uPref confID uID p paperID pref)
   thus ?thesis using Step unfolding Uact uPref isRev_def2
    by (blast dest: pref_Conflict_isRevNth reach.Step)
  qed (insert Step,
   simp add: Uact isRev_def2 u_defs pref_Conflict_isRevNth_def)+
 next
  case (UUact uua) show ?thesis using Step unfolding UUact isRev_def2
   by (meson IO_Automaton.reach.Step pref_Conflict_isRevNth)
 qed simp+
qed (simp add: istate_def)
```

Induction Rule

Simplifier

Arithmetic Procedure

General Reasoner

First-Order Provers via SLEDGEHAMMER

☐ **boilerplate**

🟥 **manual hints**

🟩 fully automatic

Application: A Verified "EasyChair"

"PC members cannot review papers if they have a conflict of interest"

Proof after Matryoshka:

```
using assms proof induction
  case (Step s a) thus ?case
  proof (cases a)
    case (Cact ca) show ?thesis
      using Step pref_Conflict_isRev reach.Step by simp
  next
    case (Uact ua) show ?thesis
    proof (cases ua)
      case (uPref confID uID p paperID pref)
      thus ?thesis

    qed (insert Step,
      simp add: Uact isRev_def2 u_defs pref_Conflict_isRevNth_def)+
  next
    case (UUact uua) show ?thesis using Step unfolding UUact isRev_def2
      by (meson IO_Automaton.reach.Step pref_Conflict_isRevNth)
  qed simp+
qed (simp add: istate_def)
```

Discover Proof Using HO Provers

⬚ missing proof

🟩 **fully automatic**

OUR GRAND CHALLENGE

Create **efficient proof calculi** and **higher-order provers** targeting proof assistants and their applications to software and hardware development

- by **fusing and extending** two lines of research: automatic proving & interactive proving

SCIENTIFIC OBJECTIVES

SO1. **Extend superposition and SMT** to higher-order logic

SO2. Design practical **methods and heuristics** based on benchmarks

SO3. Conceive **stratified architectures** to build higher-order provers

SO4. **Integrate** our provers into proof assistants (ISABELLE, LEAN, TLA$^+$)

SO1—HIGHER-ORDER SUPERPOSITION (λSUP)

First-order rule:

$$\dfrac{D' \vee t \approx t' \qquad\qquad C' \vee s[u] \not\approx s'}{(D' \vee C' \vee s[t'] \not\approx s')\sigma}\ \textsc{Sup-Left}$$

where $\sigma = \mathrm{mgu}(t, u)$ u is not a variable $t\sigma \not\precsim t'\sigma$ $s\sigma \not\precsim s'\sigma$

$(t \approx t')\sigma$ is strictly maximal in $(D' \vee t \approx t')\sigma$ and no selection

$(s \not\approx s')\sigma$ is maximal in $(C' \vee s \not\approx s')\sigma$ or selected

SO1 — HIGHER-ORDER SUPERPOSITION (λSUP)

First-order rule:

$$\frac{D' \vee t \approx t' \qquad\qquad C' \vee s[u] \not\approx s'}{(D' \vee C' \vee s[t'] \not\approx s')\sigma}\ \textsc{Sup-Left}$$

where **$\sigma = \mathrm{mgu}(t, u)$** $\quad u$ is not a variable $\quad t\sigma \not\preceq t'\sigma \quad s\sigma \not\preceq s'\sigma$

$(t \approx t')\sigma$ is strictly maximal in $(D' \vee t \approx t')\sigma$ and no selection

$(s \not\approx s')\sigma$ is maximal in $(C' \vee s \not\approx s')\sigma$ or selected

- We need **sequences of unifiers**

SO1—HIGHER-ORDER SUPERPOSITION (λSUP)

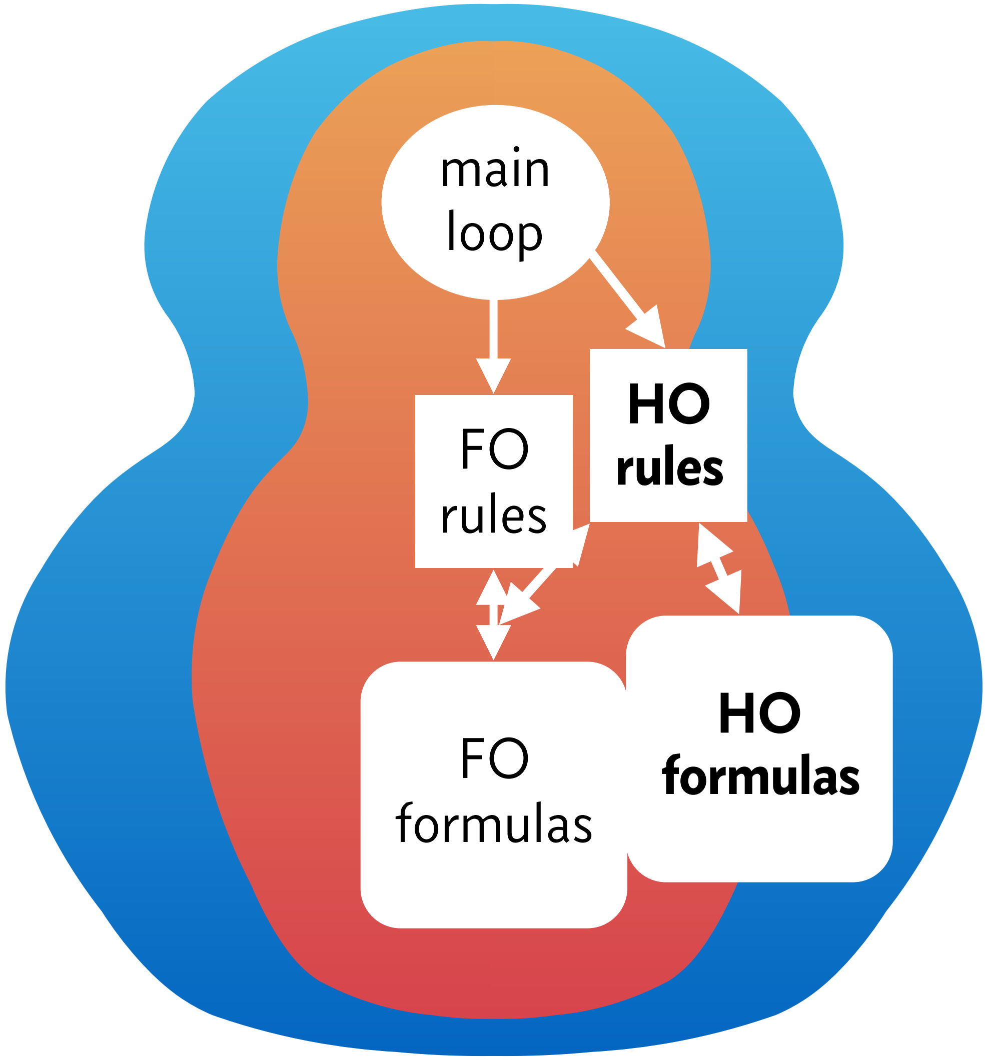
First-order rule:

$$\frac{D' \vee t \approx t' \qquad\qquad C' \vee s[u] \not\approx s'}{(D' \vee C' \vee s[t'] \not\approx s')\sigma}\ \textsc{Sup-Left}$$

where **$\sigma = \mathrm{mgu}(t, u)$** u is not a variable **$t\sigma \not\preceq t'\sigma$** **$s\sigma \not\preceq s'\sigma$**
$(t \approx t')\sigma$ is strictly maximal in $(D' \vee t \approx t')\sigma$ and no selection
$(s \not\approx s')\sigma$ is maximal in $(C' \vee s \not\approx s')\sigma$ or selected

- We need **sequences of unifiers**
- We need **higher-order term ordering**
- We also want **proof-assistant-style HO rewriting**

SO3 — STRATIFIED ARCHITECTURE



Inspired by Nelson–Oppen (SMT)

Base FO provers: E & VERIT

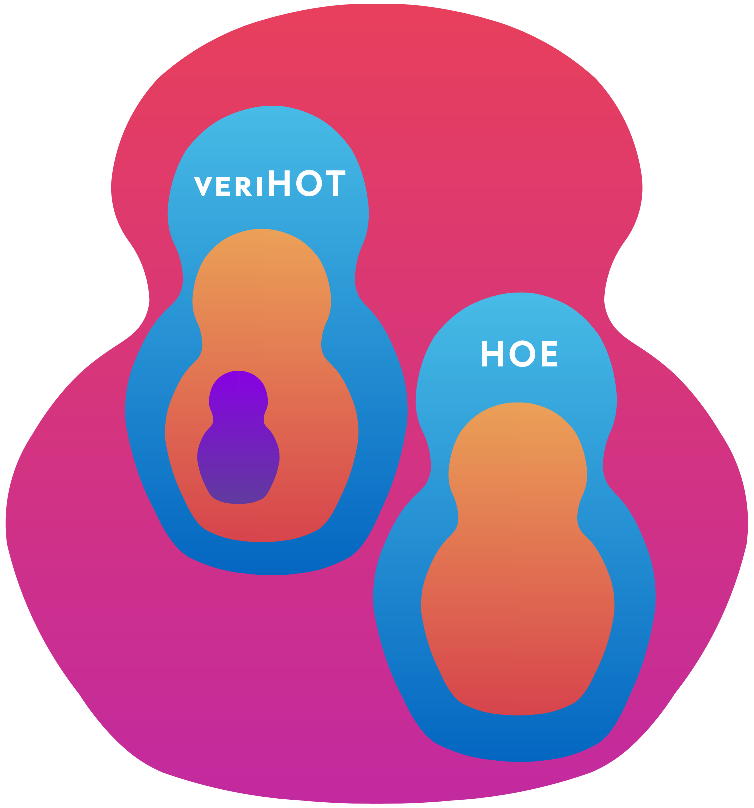
Some scientific challenges:

- How to exploit derived FO formulas and/or candidate models to guide HO quantifier instantiation?
- How to generate certificates for reconstruction in proof assistants?

First-Order Prover (e.g. VERIT)
MATRYOSHKA Prover (e.g. VERIHOT)

SO4—Connection with Proof Assistants

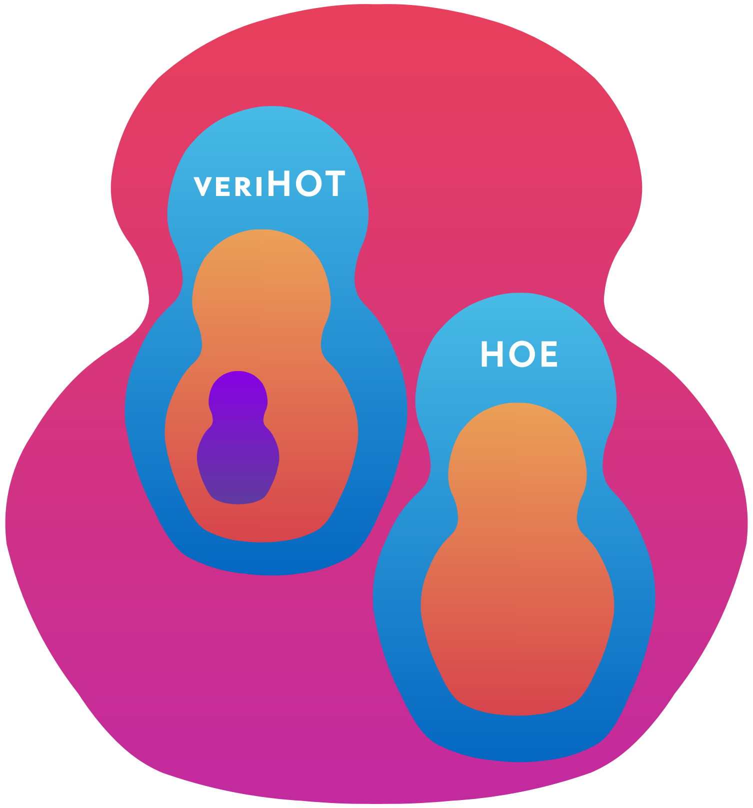
Dependent
Type Theory



Lean

Agda
Coq
Matita
⋮

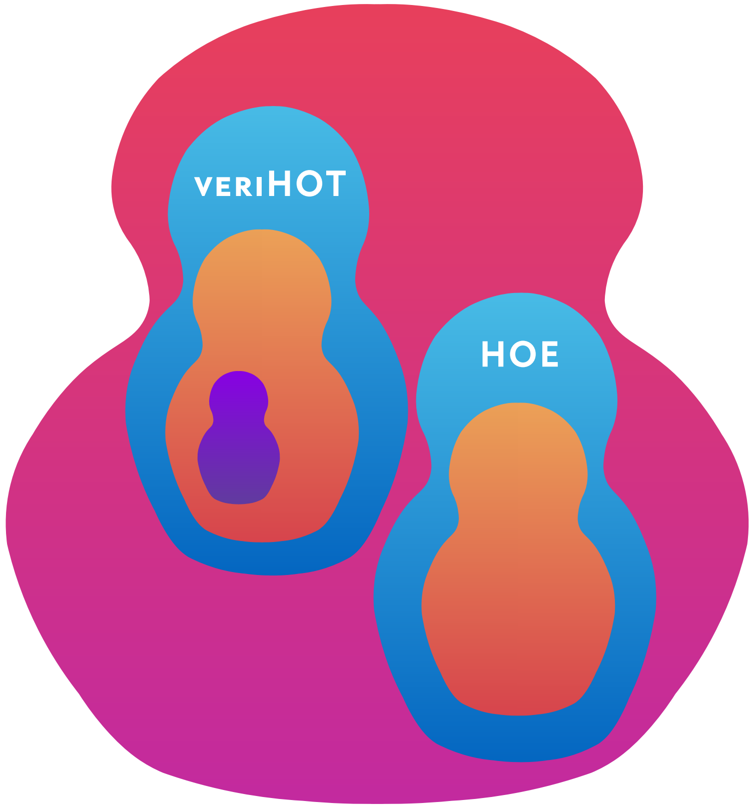
Classical
Higher-Order Logic



Isabelle/HOL

HOL4
HOL Light
PVS
⋮

Set Theory



TLA$^+$

Isabelle/ZF
Mizar
Rodin (Event-B)
⋮

THE matryoshka TEAM

Scientific Leader:	Jasmin Blanchette	*Adam*
Senior Collaborator:	Pascal Fontaine	*Ncy*
Postdoctoral Researchers:	Johannes Hölzl	*Adam*
	Rob Lewis	*Adam*
Ph.D. Students:	Alex Bentkamp	*Adam*
	Daniel El Ouraoui	*Ncy*
	Hans-Jörg Schurr	*Ncy*
	Petar Vukmirović	*Adam*

Associated Members:	Stephan Schulz	*Stgt*
	Uwe Waldmann	*SB*
Other Collaborators:	Haniel Barbosa	*Ncy*
	Simon Cruanes	*Ncy*
	Simon Robillard	*Gbg*
	& more	

http://matryoshka.gforge.inria.fr

A lot of work has gone into engineering the individual proof assistants. Maybe too little has been into developing compositional methods and tools with a **broad applicability** across systems?

Have we done enough for automated reasoning to be used as a tool, where it is needed, for real-life applications? Aren't we creating a FOL playground, whereas the **world expects HO**?