Matryoshka

Fast Interactive Verification *through* Strong Higher-Order Automation

Jasmin Christian Blanchette

Education

2000	B.Sc.	Université de Sherbrooke, Canada
2008	M.Sc.	Universitetet i Oslo, Norway
2012	Ph.D.	TU München, Germany

Employment

2000–2008	Software Engineer	Trolltech (Qt), Norway
2008–2014	Research Assistant	TU München, Germany
2015–	Senior Researcher	Inria, France

Education

2000	B.Sc.	Université de Sherbrooke, Canada
2008	M.Sc.	Universitetet i Oslo, Norway
2012	**Ph.D.**	**TU München, Germany**

Employment

2000–2008	Software Engineer	Trolltech (Qt), Norway
2008–2014	Research Assistant	TU München, Germany
2015–	Senior Researcher	Inria, France

Education

2000	B.Sc.	Université de Sherbrooke, Canada
2008	M.Sc.	Universitetet i Oslo, Norway
2012	**Ph.D.**	**TU München, Germany**

Employment

2000–2008	**Software Engineer**	**Trolltech (Qt), Norway**
2008–2014	Research Assistant	TU München, Germany
2015–	Senior Researcher	Inria, France

Education

2000	B.Sc.	Université de Sherbrooke, Canada
2008	M.Sc.	Universitetet i Oslo, Norway
2012	**Ph.D.**	**TU München, Germany**

Employment

2000–2008	**Software Engineer**	**Trolltech (Qt), Norway**
2008–2014	Research Assistant	TU München, Germany
2015–	**Senior Researcher**	**Inria, France**

Research Area

Proof and counterexample generation in proof assistants

Scientific Impact

10 articles in leading journals (e.g. *J. Autom. Reasoning*, *LMCS*)
26 papers at major conferences (e.g. ICFP, IJCAI, IJCAR, LICS)
1670+ citations on Google Scholar
Best paper award at IJCAR 2016

Professional Activities

PC chair of ITP 2016
PC member of top deduction conferences (CADE, IJCAR, …)

Current Supervision

3 Ph.D. students, 1 postdoc

Research Area

Proof and counterexample generation in proof assistants

Scientific Impact

10 articles in leading journals (e.g. *J. Autom. Reasoning*, *LMCS*)
26 papers at major conferences (e.g. ICFP, IJCAI, IJCAR, LICS)
1670+ citations on Google Scholar
Best paper award at IJCAR 2016

Professional Activities

PC chair of ITP 2016
PC member of top deduction conferences (CADE, IJCAR, …)

Current Supervision

3 Ph.D. students, 1 postdoc

Research Area

Proof and counterexample generation in proof assistants

Scientific Impact

10 articles in leading journals (e.g. *J. Autom. Reasoning*, *LMCS*)
26 papers at major conferences (e.g. ICFP, IJCAI, IJCAR, LICS)
1670+ citations on Google Scholar
Best paper award at IJCAR 2016

Professional Activities

PC chair of ITP 2016
PC member of top deduction conferences (CADE, IJCAR, …)

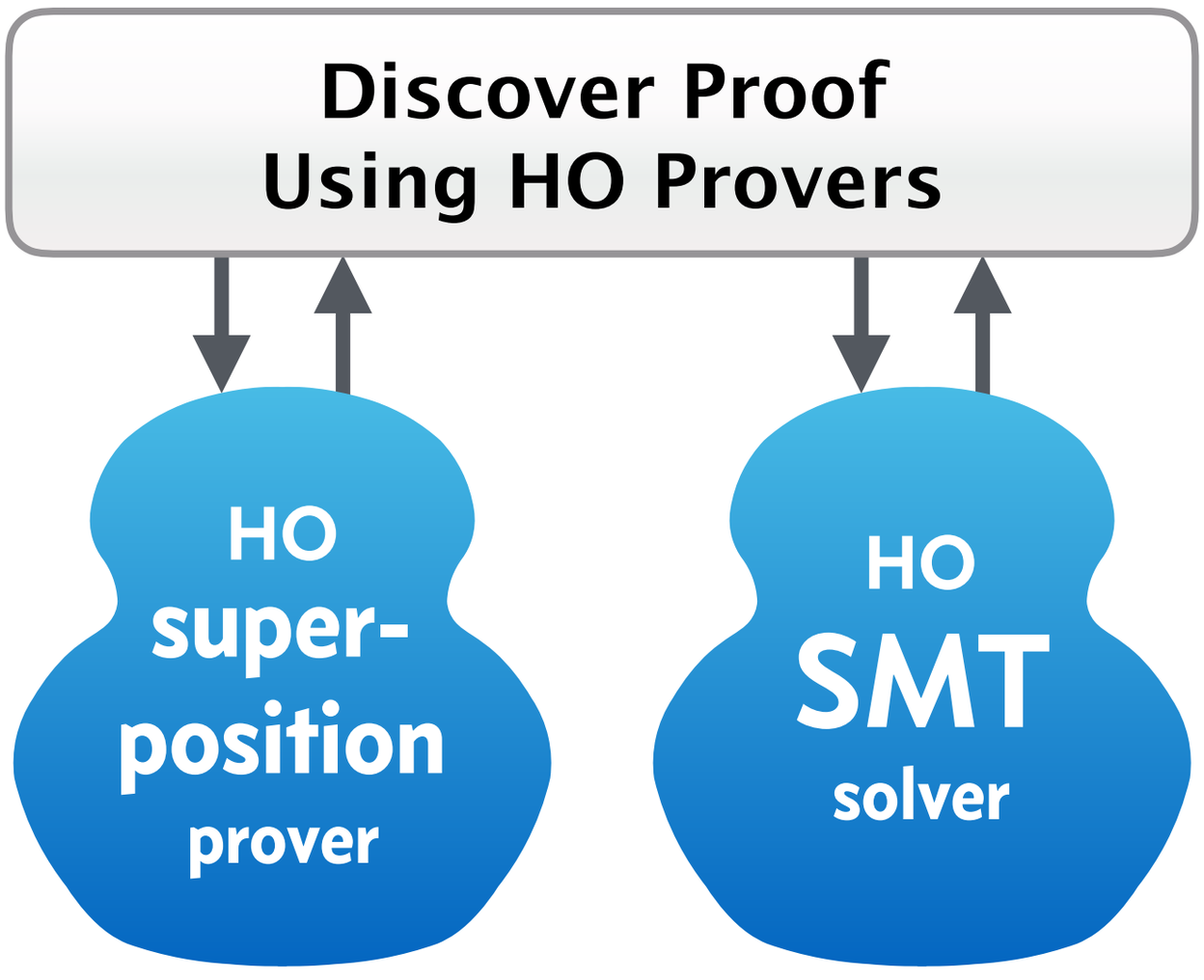
Current Supervision

3 Ph.D. students, 1 postdoc

Vision: Take the Hard Labor out of Interactive Verification

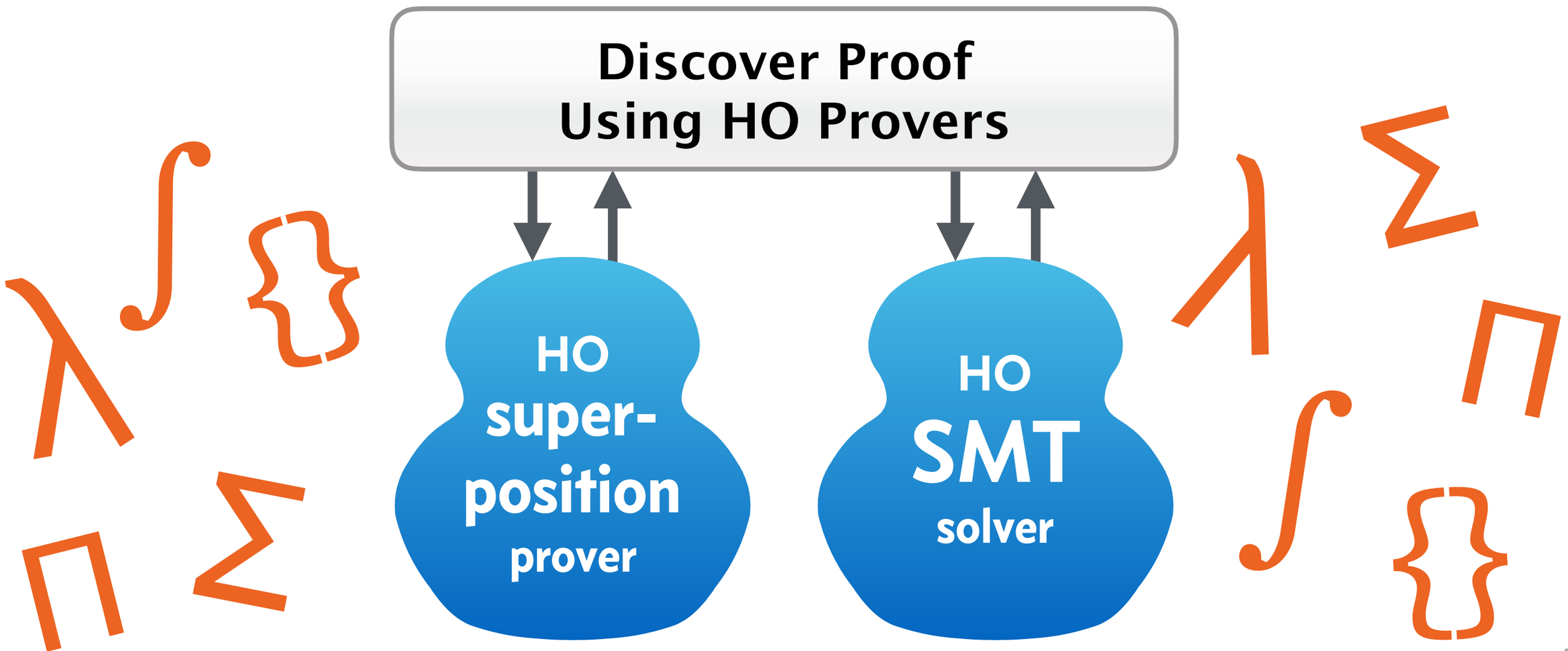
Vision: Take the Hard Labor out of Interactive Verification

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



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**



Application: A Verified "EasyChair"

- Induction Rule
- Simplifier
- Arithmetic Procedure
- General Reasoner
- First-Order Provers via Sledgehammer

Application: A Verified "EasyChair"

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

- Induction Rule
- Simplifier
- Arithmetic Procedure
- General Reasoner
- First-Order Provers via Sledgehammer

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

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

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

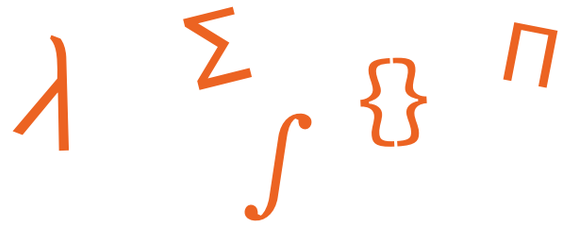
manual hints

fully automatic

Application: A Verified "EasyChair"

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

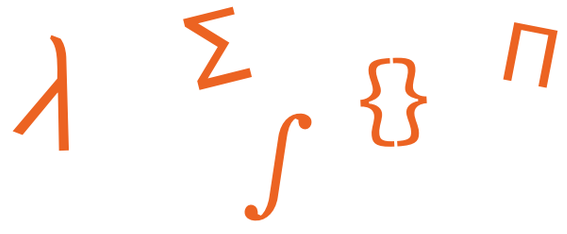
Discover Proof Using HO Provers



Application: A Verified "EasyChair"

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

Discover Proof Using HO Provers



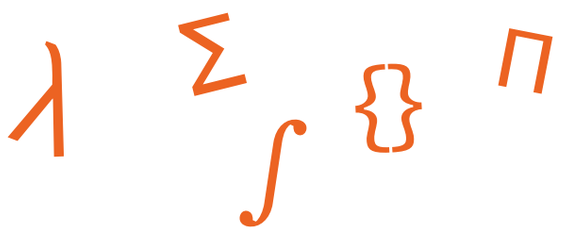
Application: A Verified "EasyChair"

"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 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)
```

Discover Proof Using HO Provers



fully automatic

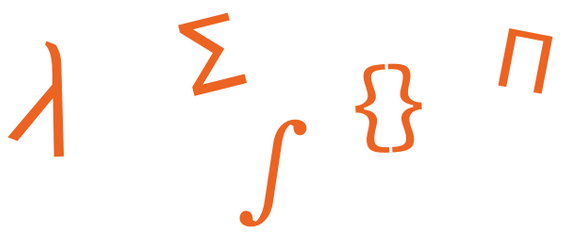
Application: A Verified "EasyChair"

"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

My Grand Challenge

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

My 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

My 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 (Coq, Isabelle, TLA$^+$)

My 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 (Coq, Isabelle, TLA$^+$)

SO1 — Higher-Order Superposition (λSUP)

First-order rule:

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

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

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

where $\sigma = \mathbf{mgu}(t, u)$ u is not a variable $t\sigma \not\trianglelefteq t'\sigma$ $s\sigma \not\trianglelefteq s'\sigma$
$(t \approx t')\sigma$ 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:

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

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**

SO1 — Higher-Order Superposition (λSUP)

First-order rule:

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

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

SO1 — Higher-Order Term Orderings

	First-order	
	KBO	LPO
Well-foundedness	✔	✔
Transitivity	✔	✔
Stability under substitution	✔	✔
FO subterm property	✔	✔
Compat. with FO contexts	✔	✔
Totality for ground terms	✔	✔

SO1 — Higher-Order Term Orderings

	First-order		Higher-order
	KBO	LPO	
Well-foundedness	✔	✔	
Transitivity	✔	✔	
Stability under substitution	✔	✔	
FO subterm property	✔	✔	
Compat. with FO contexts	✔	✔	
Totality for ground terms	✔	✔	
HO subterm property			
Compat. with HO contexts			

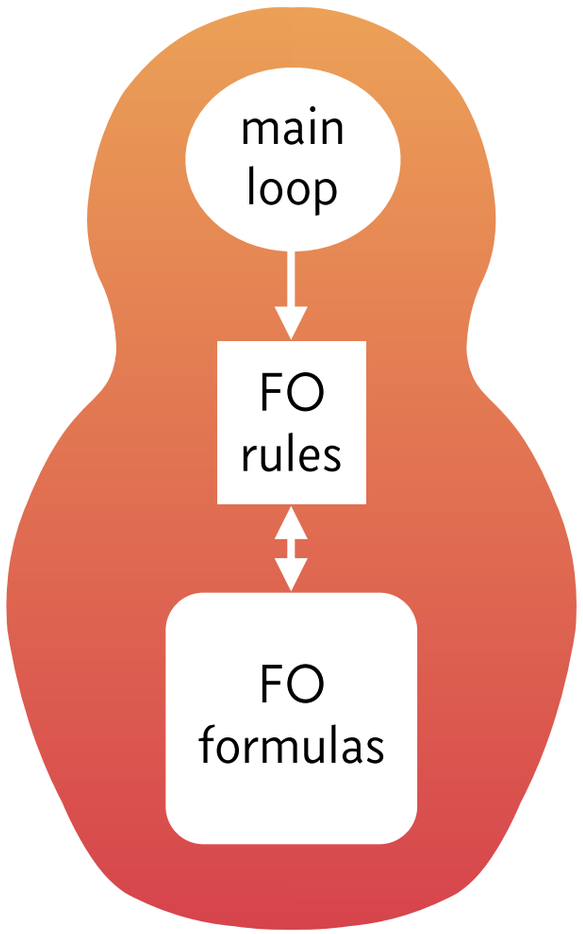
SO1 — Higher-Order Term Orderings

	First-order		Higher-order
	KBO	LPO	CPO/ HORPO
Well-foundedness	✔	✔	✔
Transitivity	✔	✔	✘
Stability under substitution	✔	✔	✘
FO subterm property	✔	✔	?
Compat. with FO contexts	✔	✔	✔
Totality for ground terms	✔	✔	✘
HO subterm property			?
Compat. with HO contexts			✔

SO1 — Higher-Order Term Orderings

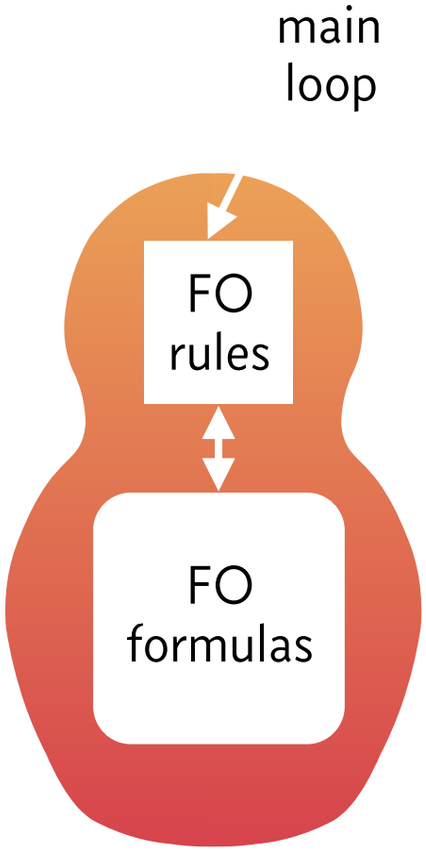
	First-order		Higher-order		
	KBO	LPO	CPO/ HORPO	λfKBO	λfLPO
Well-foundedness	✔	✔	✔	✔	✔
Transitivity	✔	✔	✘	✔	✔
Stability under substitution	✔	✔	✘	✔	✔
FO subterm property	✔	✔	?	✔	✔
Compat. with FO contexts	✔	✔	✔	✔	✔
Totality for ground terms	✔	✔	✘	✔	✔
HO subterm property			?	✔	✔
Compat. with HO contexts			✔	✔	(✔)

SO3—Stratified Architecture



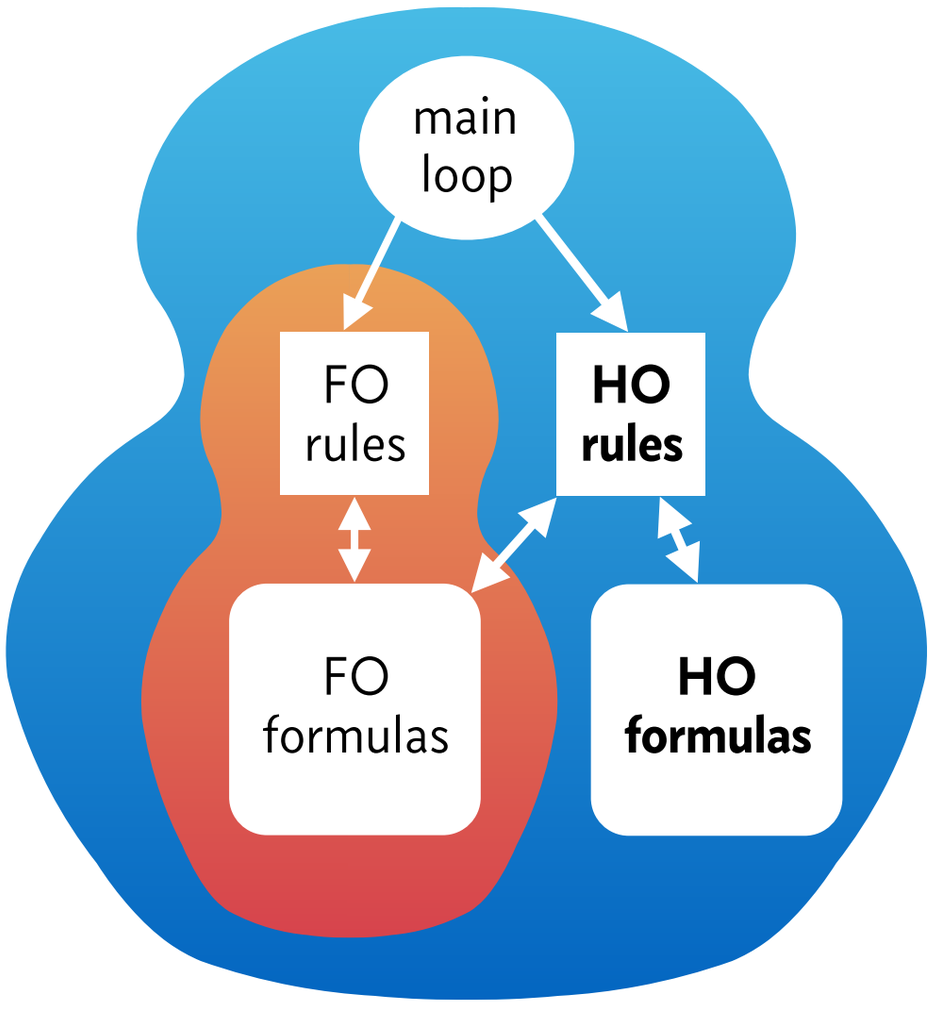
First-Order Prover (e.g. veriT)

SO3 — Stratified Architecture



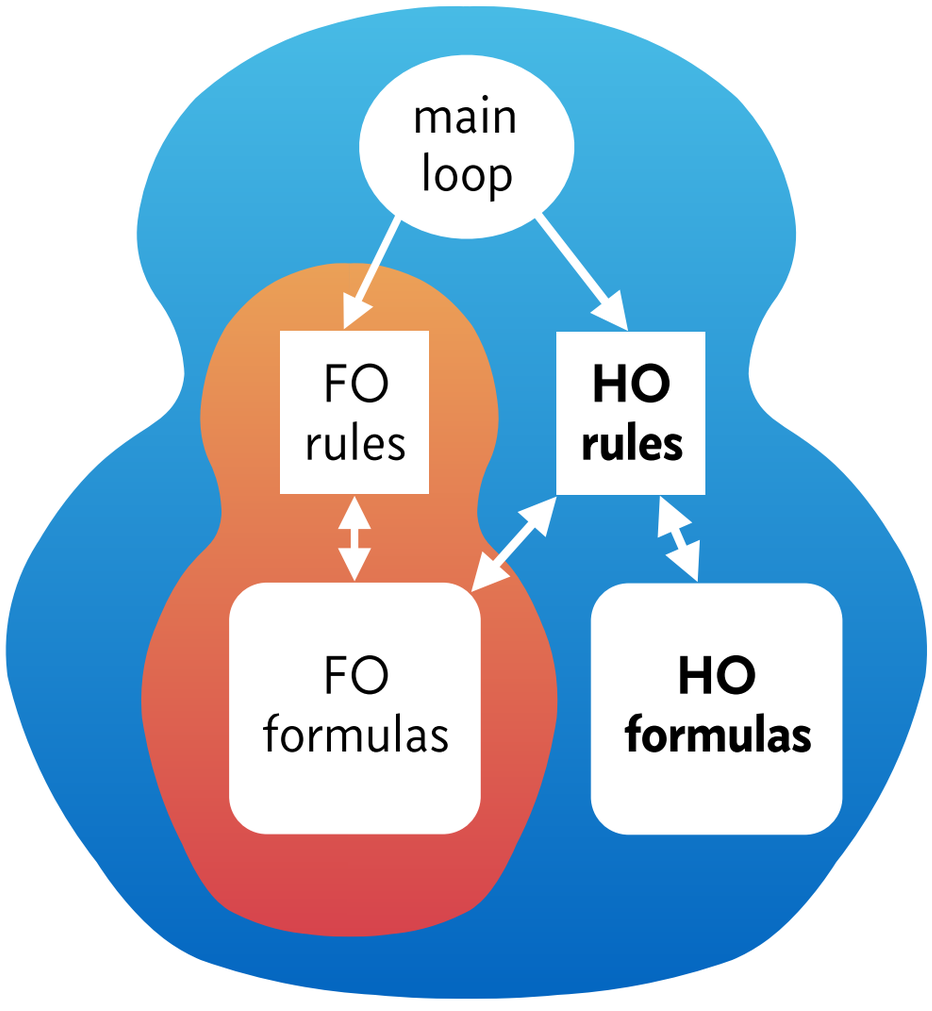
First-Order Prover (e.g. veriT)

SO3—Stratified Architecture



First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)

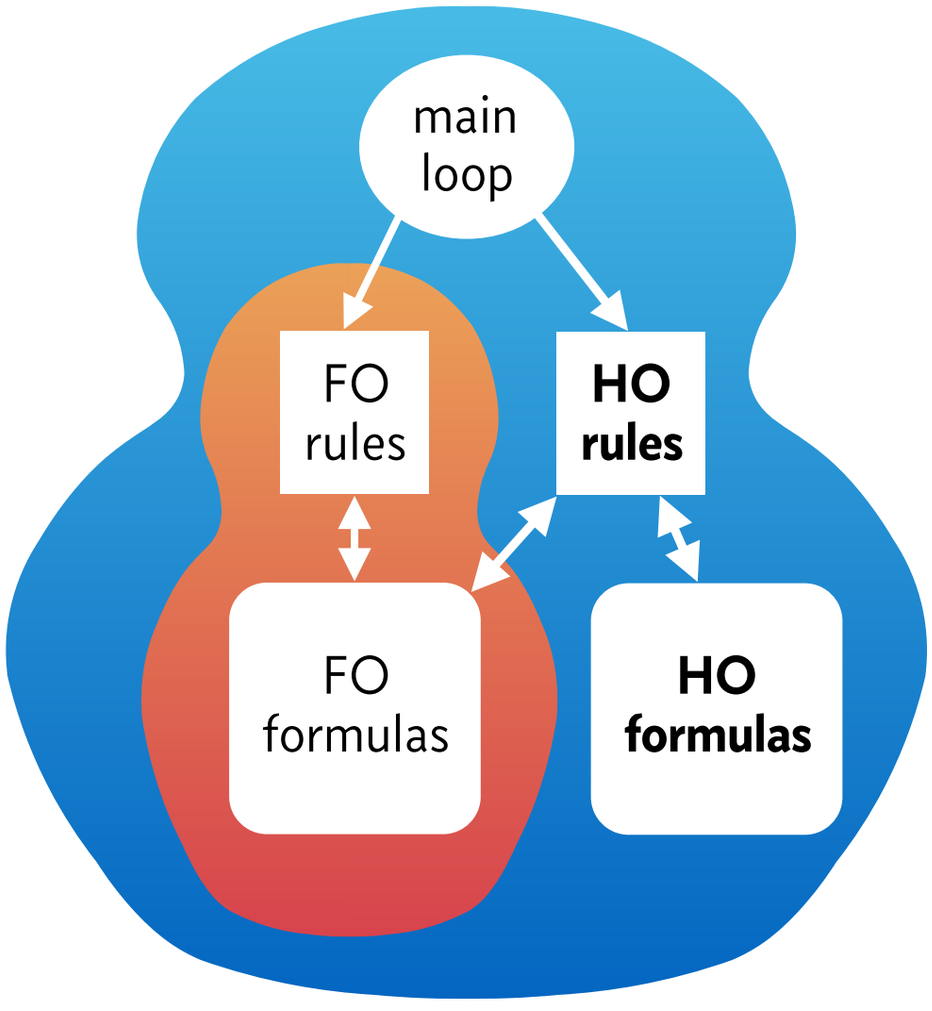
SO3—Stratified Architecture



Inspired by Nelson–Oppen (SMT)

First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)

SO3—Stratified Architecture

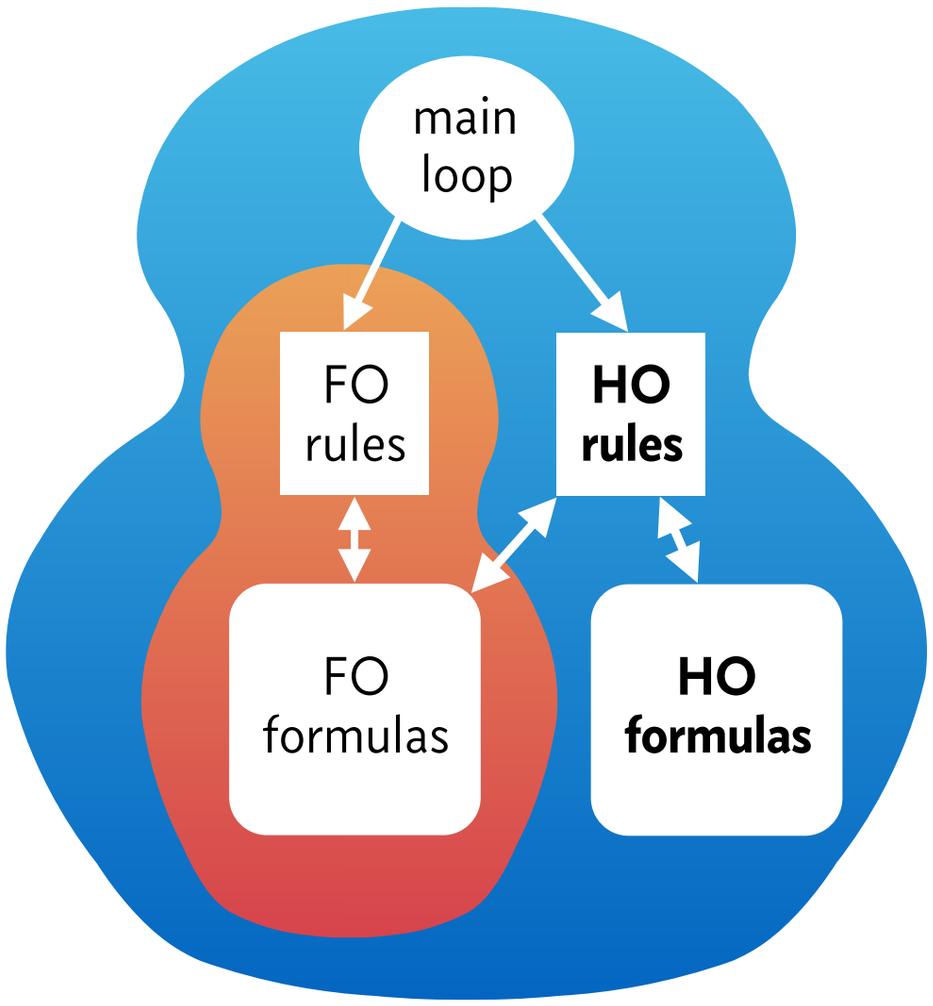


Inspired by Nelson–Oppen (SMT)

Base FO provers: SPASS & veriT

First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)

SO3—Stratified Architecture



Inspired by Nelson–Oppen (SMT)

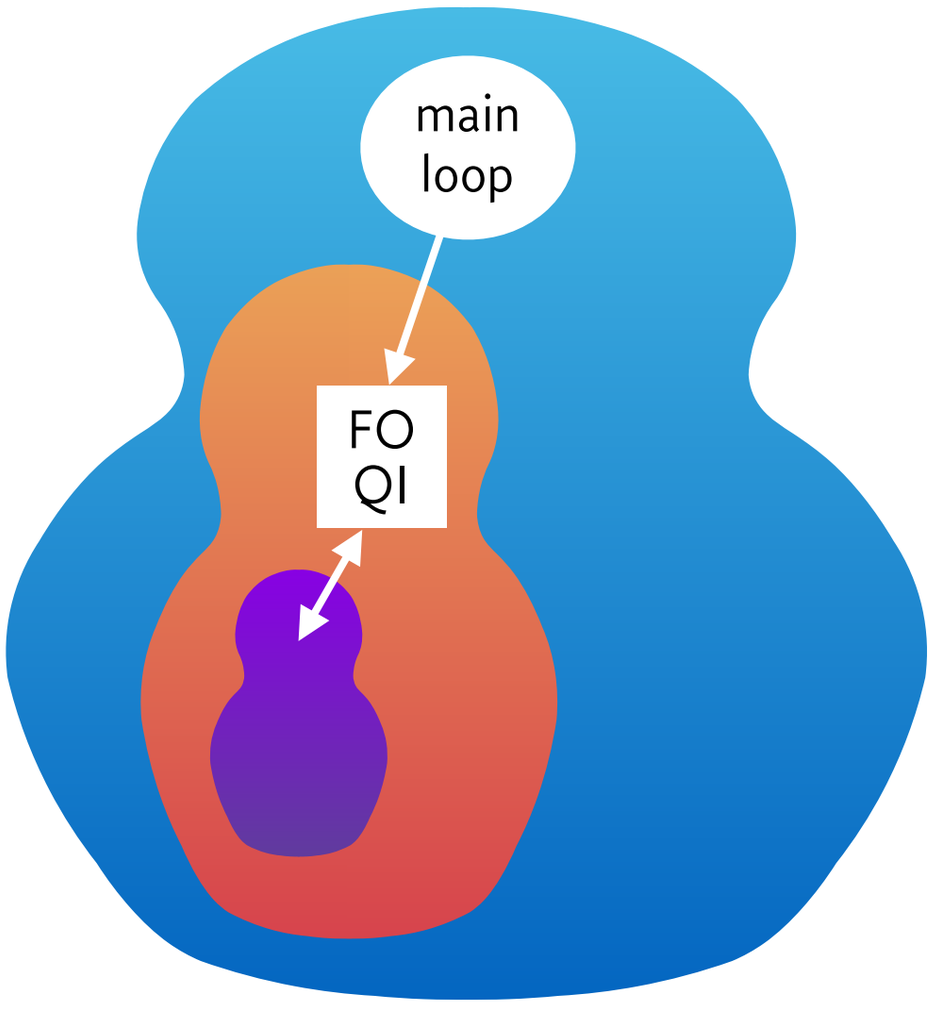
Base FO provers: SPASS & veriT

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)

SO3 — Higher-Order SMT

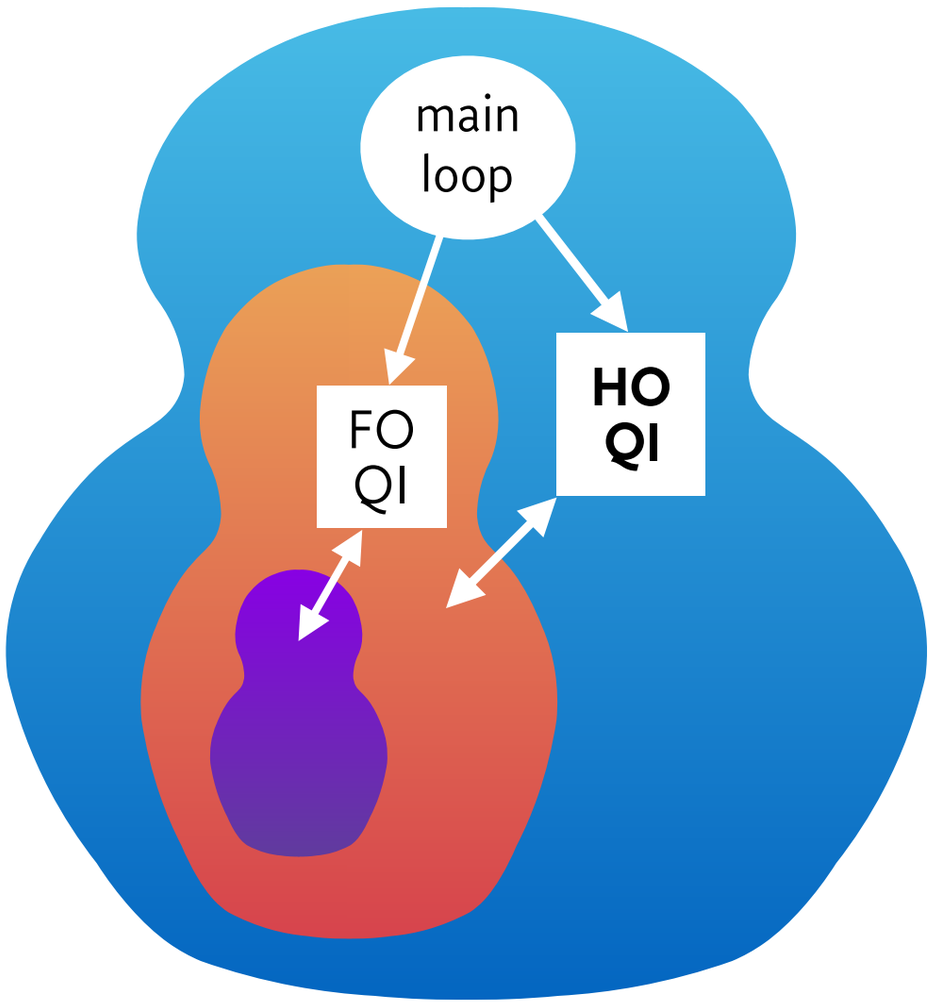


First-order quantifier instantiation (QI):

- E-matching (triggers)
- Model-based QI
- Conflict-guided instantiation
- Congruence closure with free variables

SAT Solver (e.g. MiniSat)
First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)

SO3 — Higher-Order SMT



First-order quantifier instantiation (QI):

- E-matching (triggers)
- Model-based QI
- Conflict-guided instantiation
- Congruence closure with free variables

- We need to **extend these strategies to higher-order logic**

SAT Solver (e.g. MiniSat)
First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)

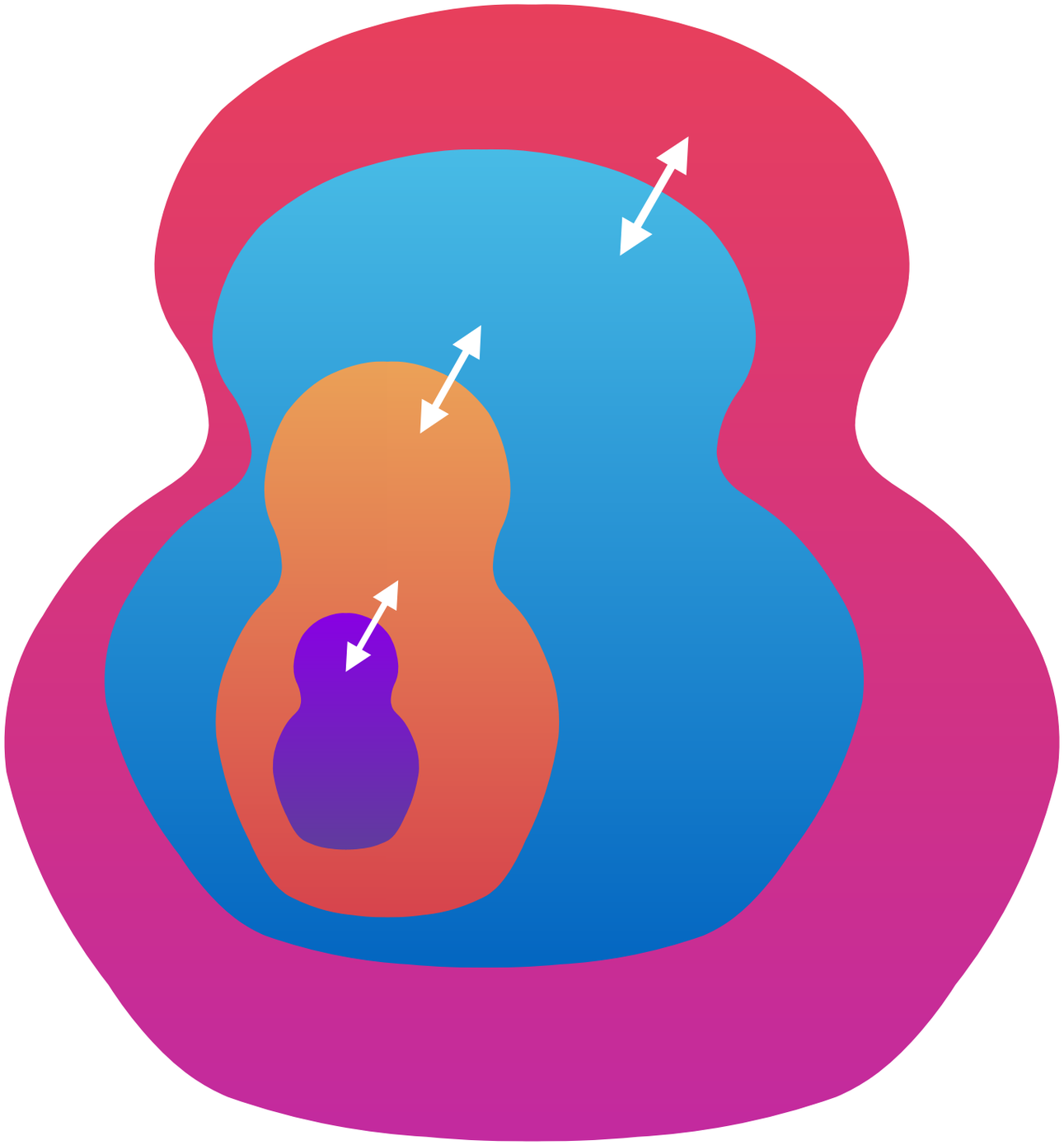
SO4—Connection with Proof Assistants

SAT Solver (e.g. MiniSat)
First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)

SO4—Connection with Proof Assistants

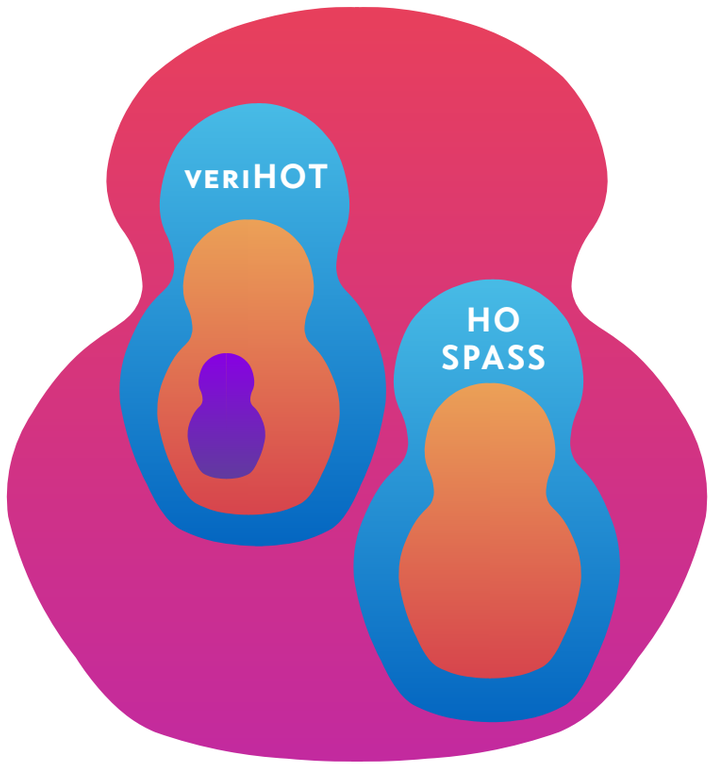
SAT Solver (e.g. MiniSat)
First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)
Proof Assistant (e.g. Coq)

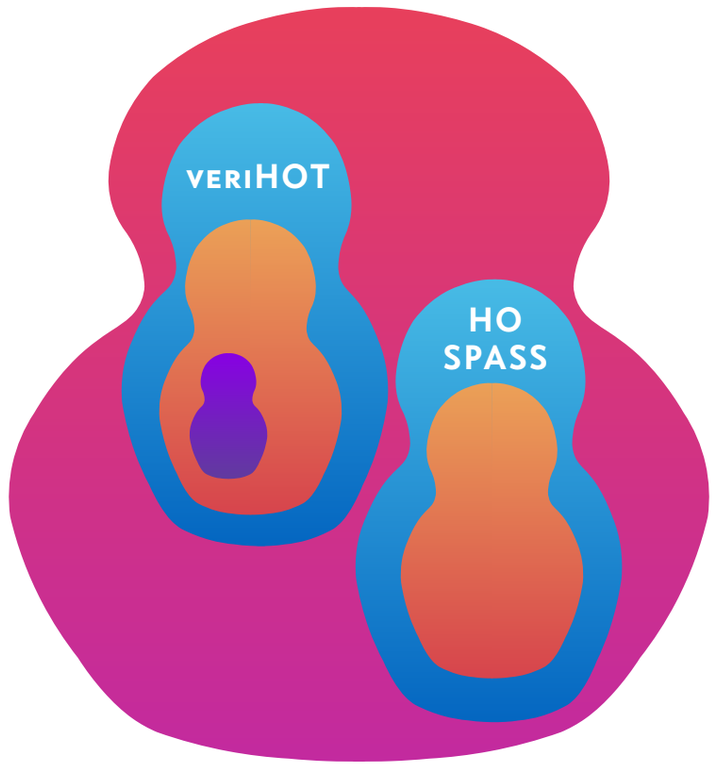
SO4—Connection with Proof Assistants

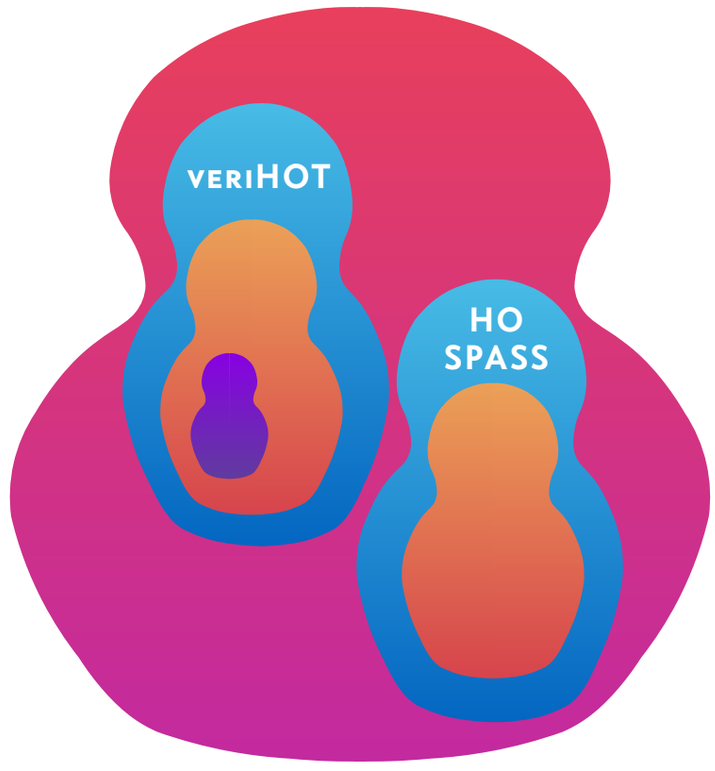


SAT Solver (e.g. MiniSat)
First-Order Prover (e.g. veriT)
Matryoshka Prover (e.g. veriHOT)
Proof Assistant (e.g. Coq)

SO4—Connection with Proof Assistants

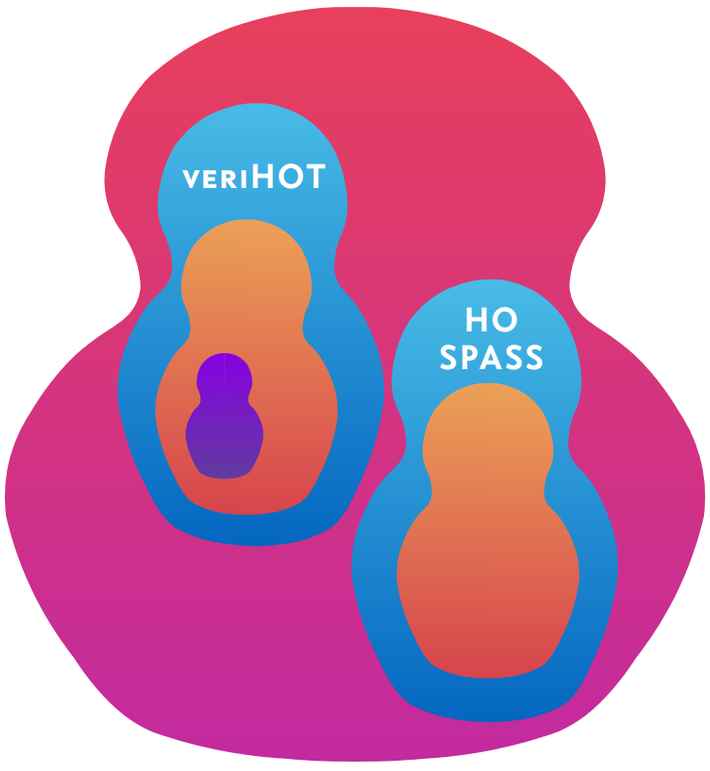






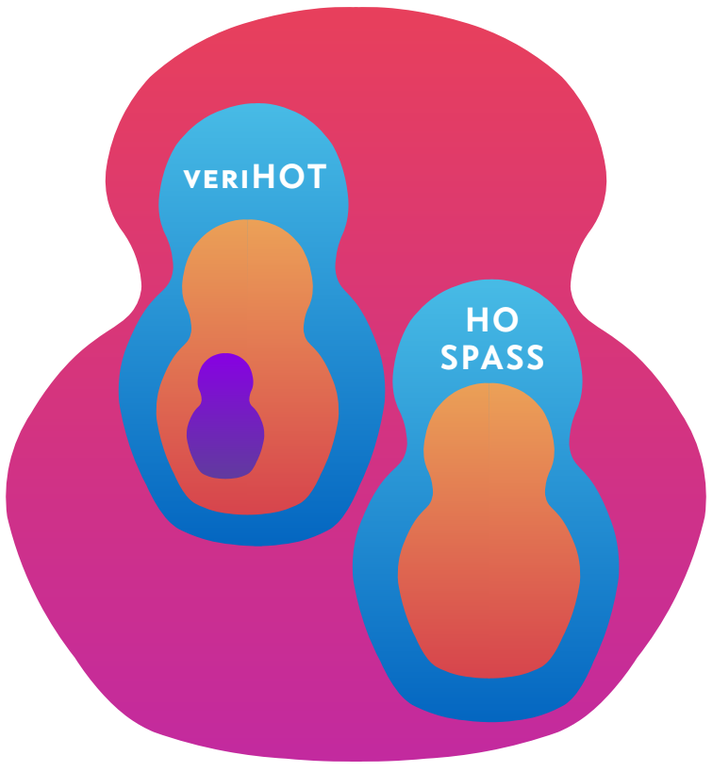
SO4—Connection with Proof Assistants

Dependent
Type Theory



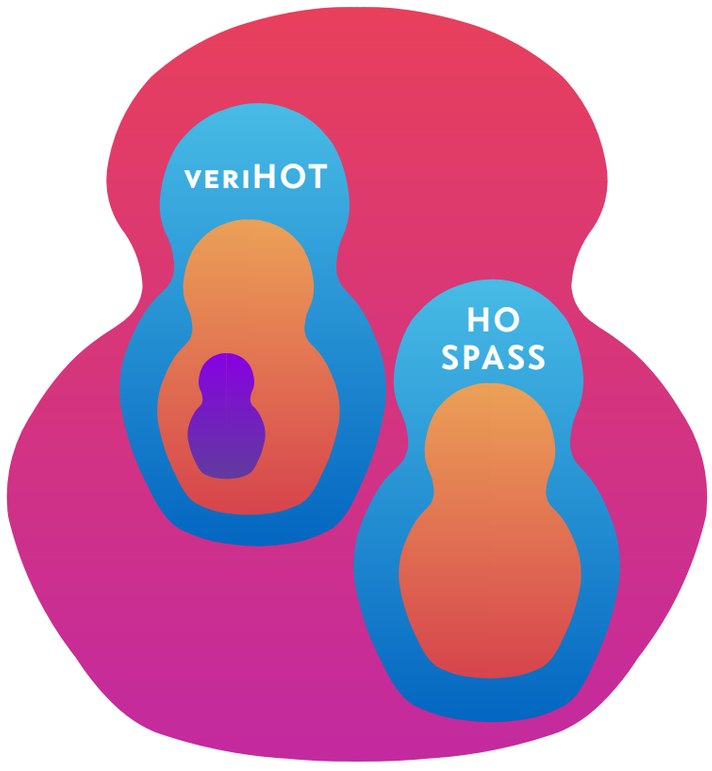
Coq

Classical
Higher-Order Logic



Isabelle/HOL

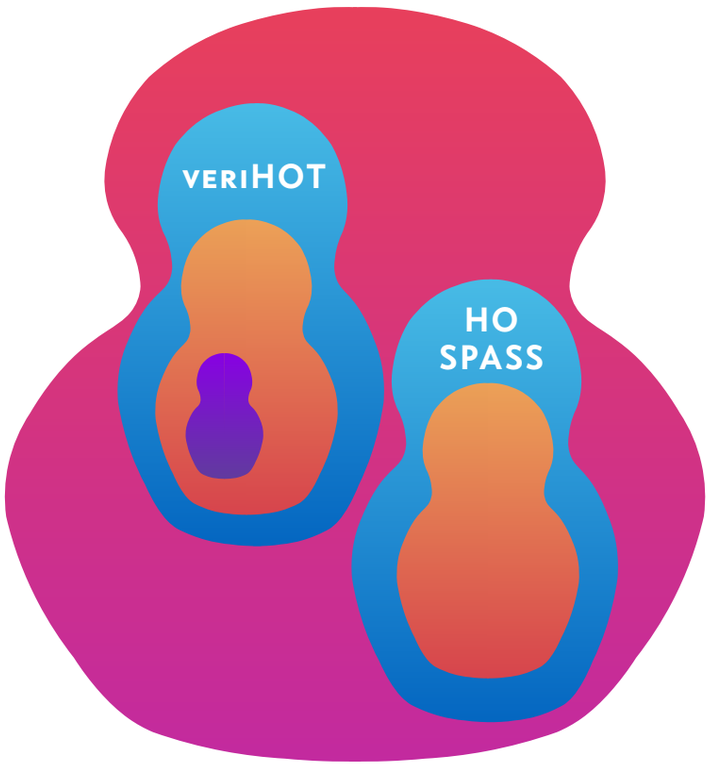
Set Theory



TLA+

SO4—Connection with Proof Assistants

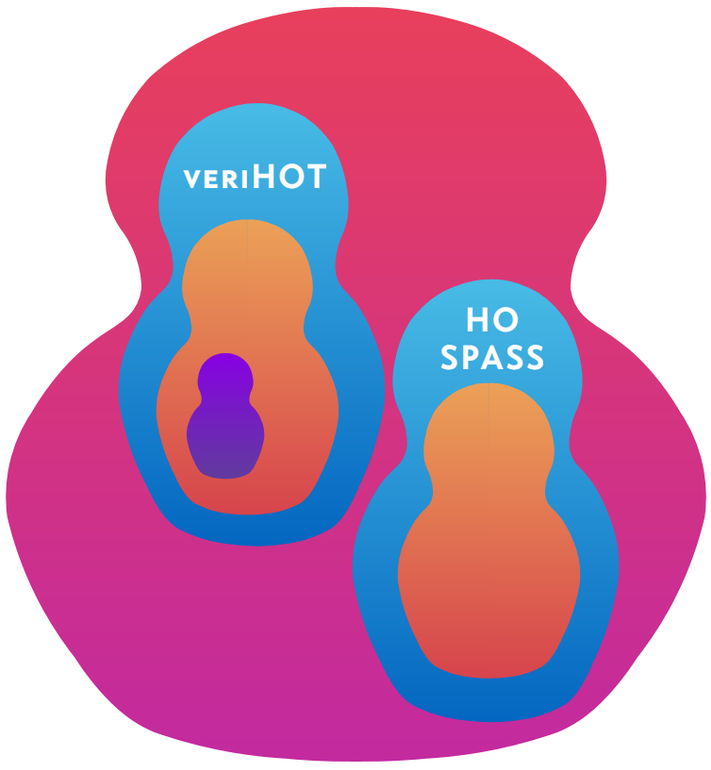
Dependent
Type Theory



Coq

Agda
Lean
Matita
⋮

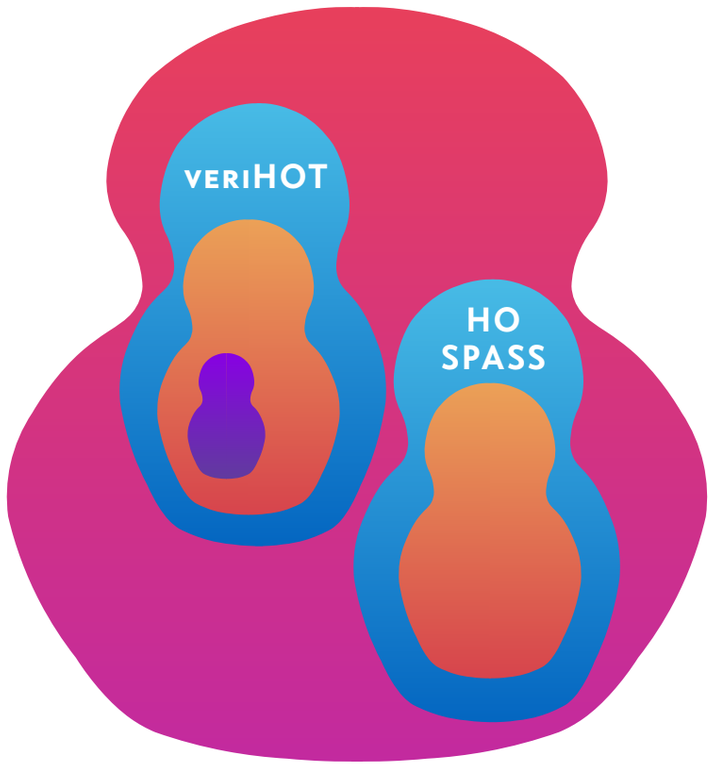
Classical
Higher-Order Logic



Isabelle/HOL

HOL4
HOL Light
PVS
⋮

Set Theory



TLA+

Isabelle/ZF
Mizar
Rodin (Event-B)
⋮

The Matryoshka Research Group

Scientific Leader: J. C. Blanchette (75%)

SMT & veriT — *Permanent Member:* Pascal Fontaine (20%)

Higher-order reasoning — Junior Researcher

Superposition — Ph.D. Student 1

SMT — Ph.D. Student 2

Proof assistants — Ph.D. Student 3

Implementation — Software Engineer

Superposition & SPASS — *External Collaborator:* Uwe Waldmann (MPII)

Visiting Researchers:

- developers of automatic provers
- developers and users of proof assistants

Matryoshka in one Slide

Grand Challenge & Outcome

Create efficient **proof calculi** (λSUP & λSMT) and stratified **higher-order provers** (HO SPASS & veriHOT) to dramatically **improve automation** in proof assistants

Risks

- Efficient HO automation is a long-standing open problem
- Proposed stratified architecture has never been tried

Impact

- The project will recast the **methods of automatic proving** to reach the **goals of interactive proving**
- Interactive verification will become a **cost-effective** option for building **software & hardware** of the **highest quality**