Mechanizing the Metatheory of Sledgehammer

Jasmin Blanchette and Andrei Popescu

Fakultät für Informatik
Technische Universität München

Overview

- Sledgehammer
- Sledgehammer's Metatheory
- Mechanization

Overview

- Sledgehammer
- Sledgehammer's Metatheory
- Formal Verification

Overview

- Sledgehammer
- Sledgehammer's Metatheory
- Formal Verification

Sledgehammer

Sledgehammer

- Isabelle: interactive theorem prover

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\, \exists y.\, P(x, y)$

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
    Fix x

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
    Fix x
    Obtain z where $Q(x, z)$ using ⟨facts⟩ by ⟨method⟩

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
  Fix x
  Obtain z where $Q(x, z)$ using ⟨facts⟩ by ⟨method⟩
  From this obtain y where $P(x, y)$

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
  Fix x
  Obtain z where $Q(x, z)$ using ⟨facts⟩ by ⟨method⟩
  From this obtain y where $P(x, y)$ call sledgehammer

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
  Fix x
  Obtain z where $Q(x, z)$ using ⟨facts⟩ by ⟨method⟩
  From this obtain y where $P(x, y)$ by metis ⟨facts⟩

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
  Fix x
  Obtain z where $Q(x, z)$ using ⟨facts⟩ by ⟨method⟩
  From this obtain y where $P(x, y)$ by metis ⟨facts⟩
qed

Sledgehammer

- Isabelle: interactive theorem prover
- Sledgehammer: connects Isabelle with automatic theorem provers (ATPs)

Lemma: $\forall x.\ \exists y.\ P(x, y)$
Proof (with Isabelle's structured proof language):
  Fix x
  Obtain z where $Q(x, z)$ using ⟨facts⟩ by ⟨method⟩
  From this obtain y where $P(x, y)$ call sledgehammer
qed

Sledgehammer's Pipeline

φ

- Start with (typed) HOL goal φ

Sledgehammer's Pipeline

$$(\bigwedge F) \stackrel{?}{\Longrightarrow} \varphi$$

- Start with (typed) HOL goal φ
- Gather set of relevant facts F

Sledgehammer's Pipeline

$$(\bigwedge F^{\#}) \stackrel{?}{\Longrightarrow} \varphi^{\#}$$

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL

Sledgehammer's Pipeline

$$(\bigwedge F^{\#}) \stackrel{?}{\Longrightarrow} \varphi^{\#}$$

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away

Sledgehammer's Pipeline

$$(\bigwedge F^{\#}) \stackrel{\cdots}{\Longrightarrow} \varphi^{\#}$$

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away
- Feed to ATPs

Sledgehammer's Pipeline

$$(\bigwedge E^{\#}) \overset{!!!}{\Longrightarrow} \varphi^{\#}$$

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away
- Feed to ATPs
- If one succeeds, with $E \subseteq F$

Sledgehammer's Pipeline

$$\left(\bigwedge E\right) \overset{\text{Isabelle}}{\Longrightarrow} \varphi$$

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away
- Feed to ATPs
- If one succeeds, with $E \subseteq F$
- Reprove with Isabelle's internal ATP, Metis

Sledgehammer's Pipeline

have φ by (metis E)

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away
- Feed to ATPs
- If one succeeds, with $E \subseteq F$
- Reprove with Isabelle's internal ATP, Metis

Sledgehammer's Pipeline

have φ by (metis E)

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away
- Feed to ATPs
- If one succeeds, with $E \subseteq F$
- Reprove with Isabelle's internal ATP, Metis
- Why reprove?

Sledgehammer's Pipeline

have φ by (metis E)

- Start with (typed) HOL goal φ
- Gather set of relevant facts F
- Encode in (untyped) FOL
  λ's go away, types go away
- Feed to ATPs
- If one succeeds, with $E \subseteq F$
- Reprove with Isabelle's internal ATP, Metis
- Why reprove? Isabelle trusts no one but herself

Overview

- Sledgehammer
- Sledgehammer's Metatheory
- Formal Verification

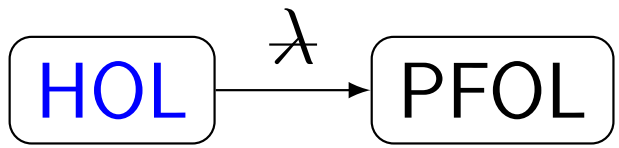
Sledgehammer's Metatheory

Higher Order Logic

HOL

Sledgehammer's Metatheory

Polymorphic First Order Logic

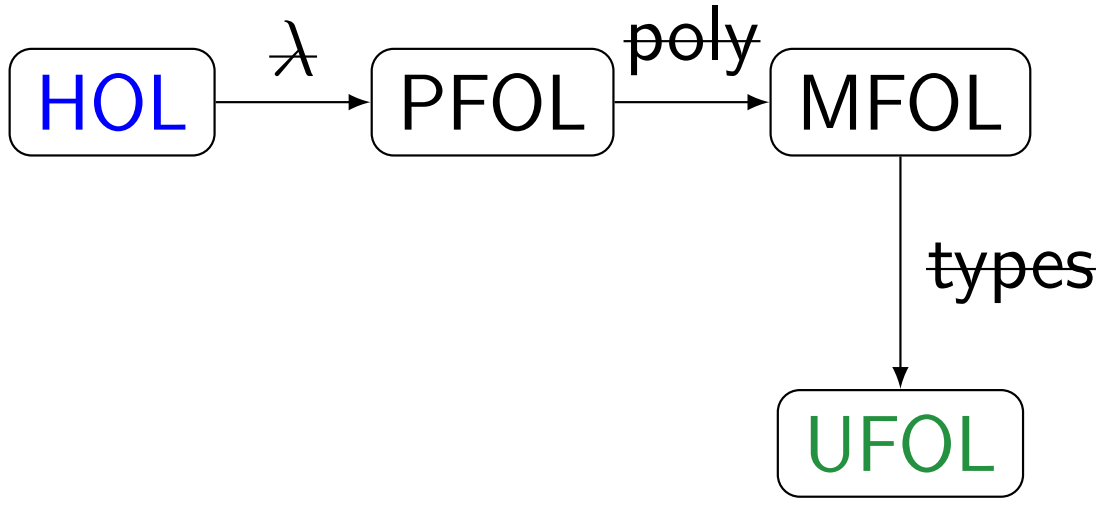


Sledgehammer's Metatheory

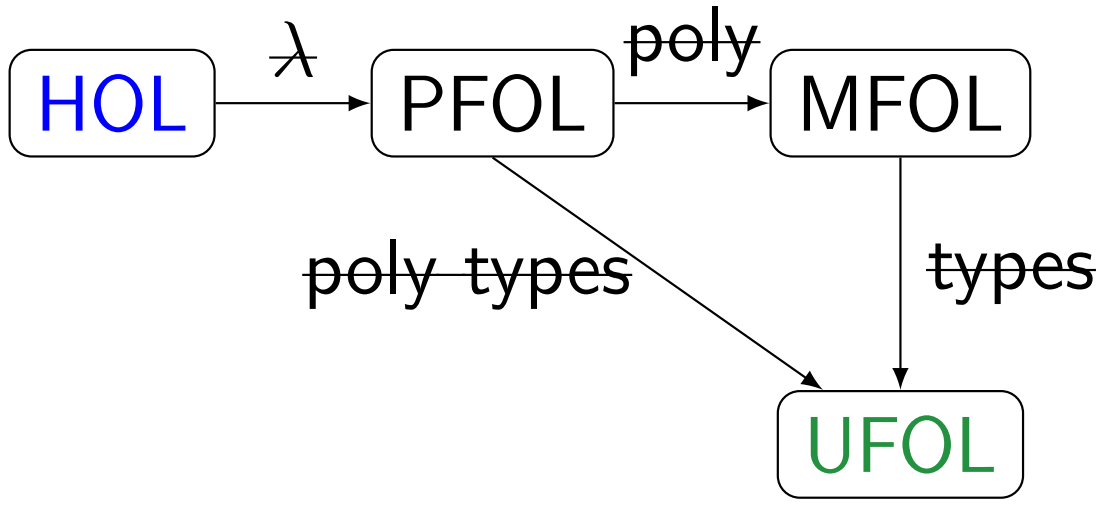
Many-Typed (Many-Sorted) First Order Logic

Sledgehammer's Metatheory

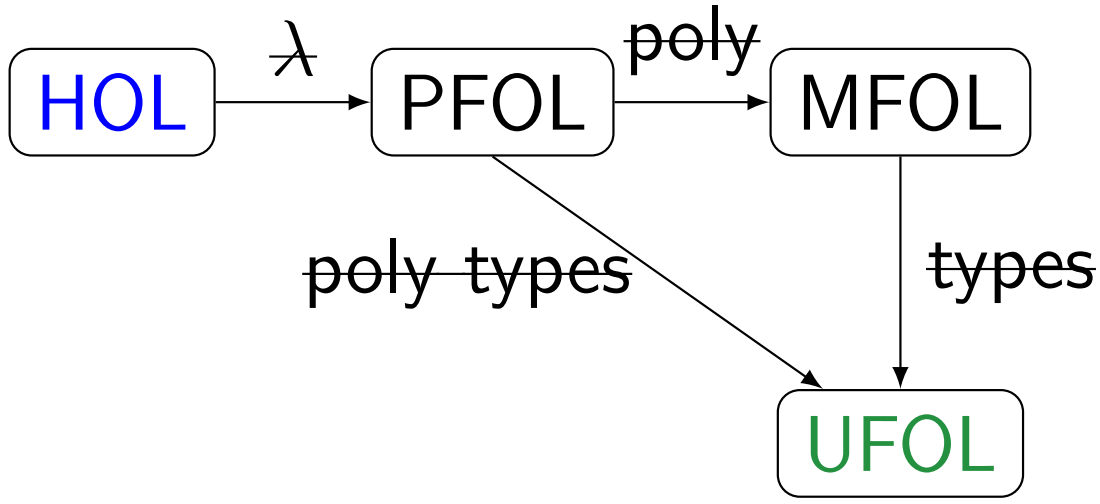
Untyped First Order Logic



Sledgehammer's Metatheory

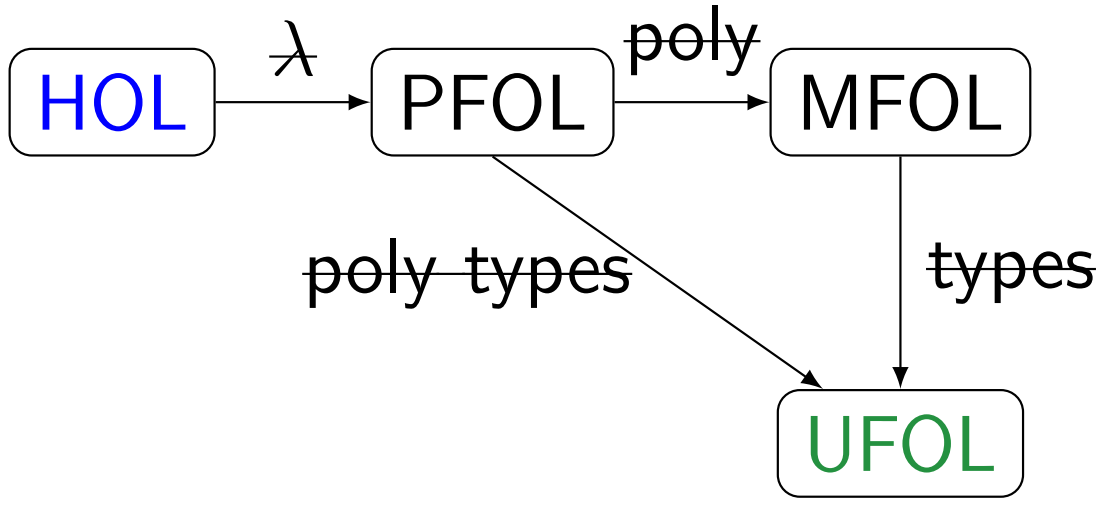


Sledgehammer's Metatheory



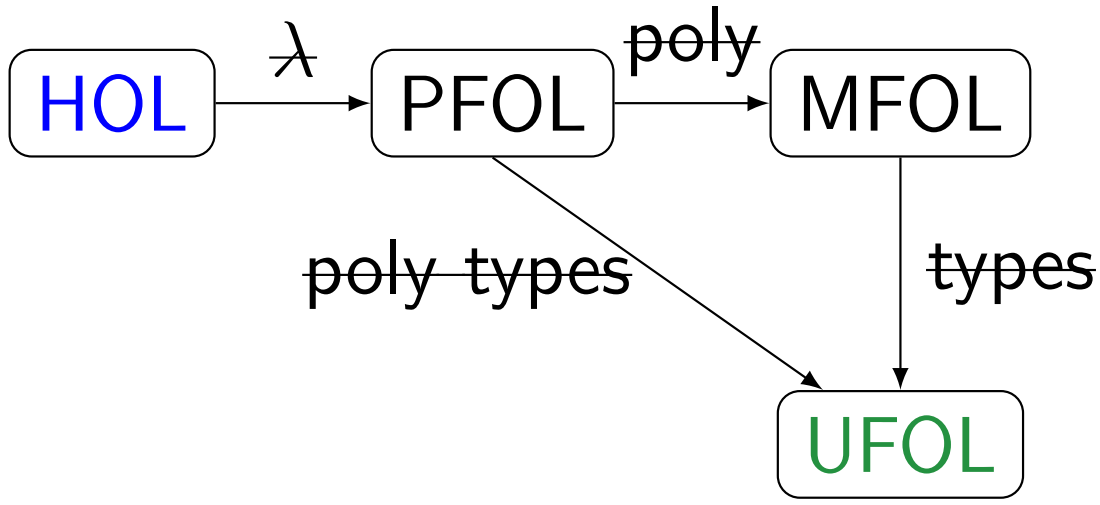
Are the encodings

Sledgehammer's Metatheory



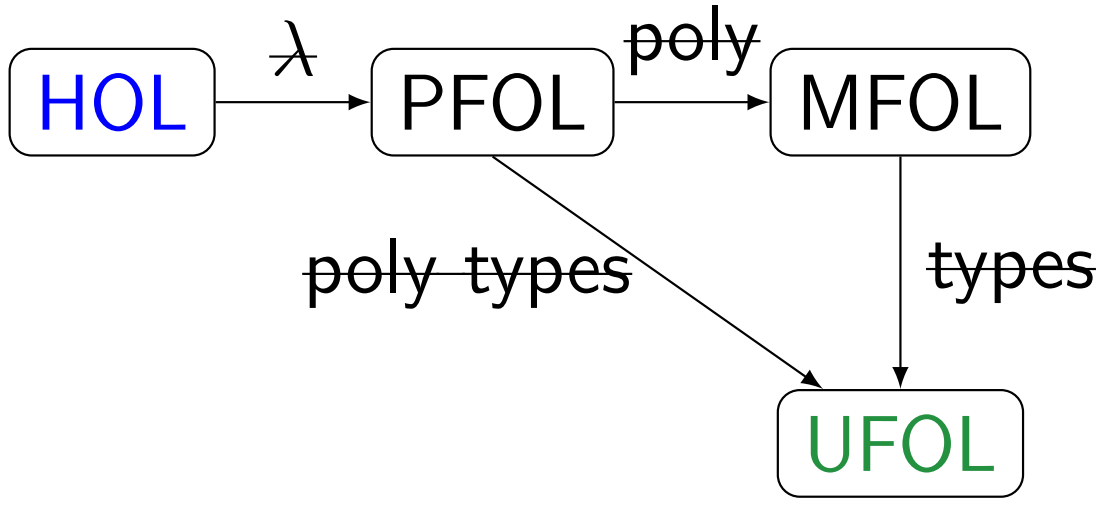
Are the encodings sound?

Sledgehammer's Metatheory



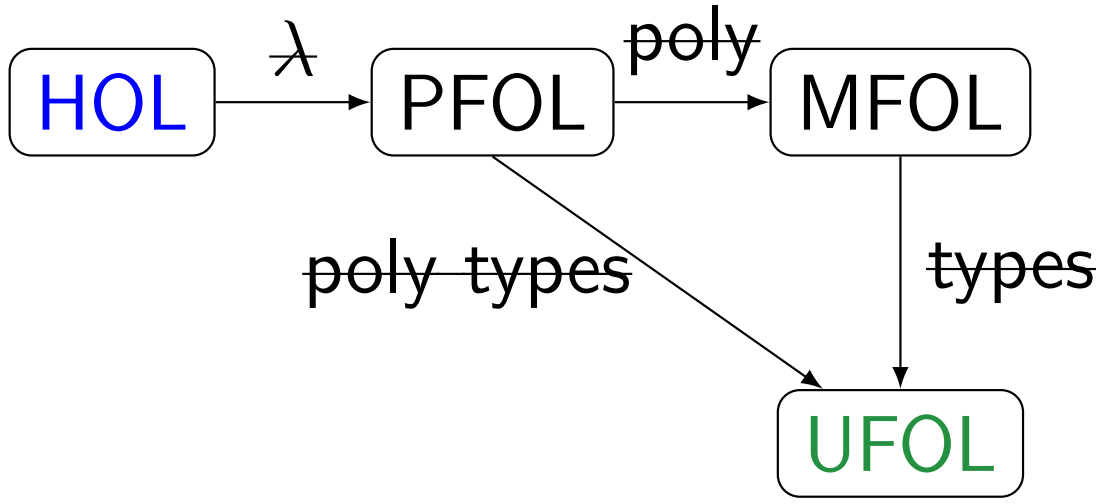
Are the encodings sound? complete?

Sledgehammer's Metatheory



Are the encodings sound? complete? efficient?

Sledgehammer's Metatheory



Are the encodings sound? complete? efficient?
All these are important practical questions!

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable
  $\mapsto$ Isabelle can reconstruct the proof

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable
  $\mapsto$ Isabelle can reconstruct the proof
- Efficiency

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable
  $\mapsto$ Isabelle can reconstruct the proof
- Efficiency
  - When erasing types, extra structure is added in compensation: tag or guard decoration

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable
  $\mapsto$ Isabelle can reconstruct the proof
- Efficiency
  - When erasing types, extra structure is added in compensation: tag or guard decoration
  - Decoration should be lightweight, yet preserve soundness and completeness

Sledgehammer's Metatheory

- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable
  $\mapsto$ Isabelle can reconstruct the proof
- Efficiency
  - When erasing types, extra structure is added in compensation: tag or guard decoration
  - Decoration should be lightweight, yet preserve soundness and completeness
    $\mapsto$ Smaller clutter means better chance for ATPs

Sledgehammer's Metatheory

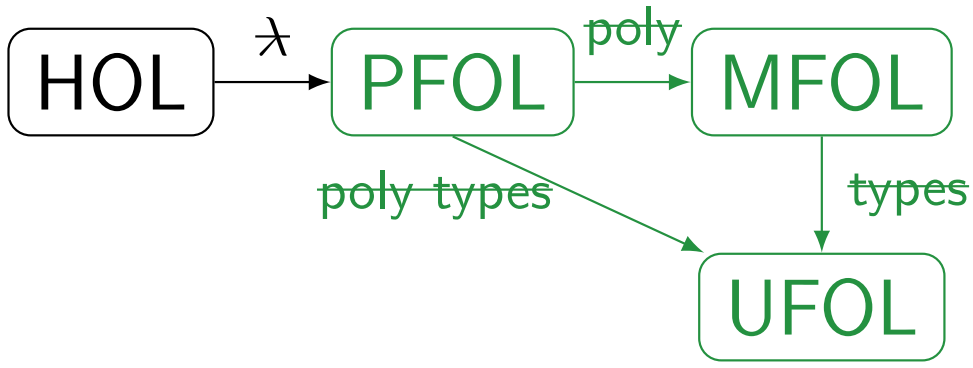
- Completeness: The orginal Isabelle goal provable implies FOL encoding also provable
  $\mapsto$ The ATPs have a chance to prove it
- Soundness: FOL encoding provable implies original Isabelle goal also provable
  $\mapsto$ Isabelle can reconstruct the proof
- Efficiency
  - When erasing types, extra structure is added in compensation: tag or guard decoration
  - Decoration should be lightweight, yet preserve soundness and completeness
    $\mapsto$ Smaller clutter means better chance for ATPs

It's all for the user ☺

Overview

- Sledgehammer
- Sledgehammer's Metatheory
- Formal Verification

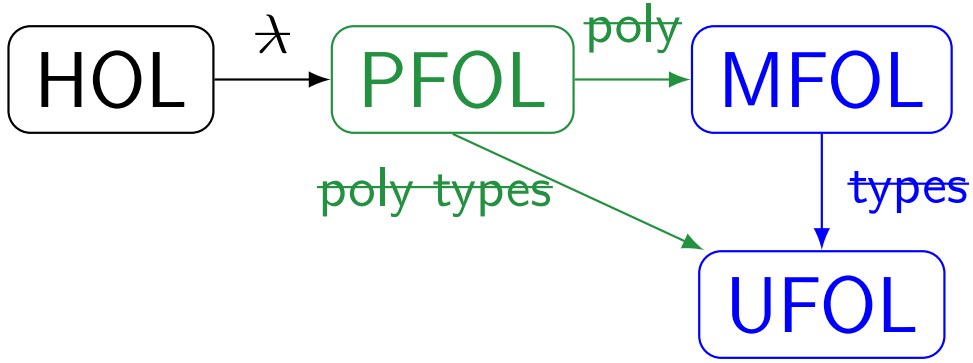
Formal Verification



Blanchette et al., TACAS 2013:
Encoding Monomorphic and Polymorphic Types
(Soundly, Completely, and Efficiently)
A variety of encodings

- based on tags or guards
- full, lightweight, featherweight
- factor in infiniteness knowledge

Formal Verification



Blanchette et al., TACAS 2013:
Encoding Monomorphic and Polymorphic Types
(Soundly, Completely, and Efficiently)
A variety of encodings

- based on tags or guards
- full, lightweight, featherweight
- factor in infiniteness knowledge

FroCoS 2013: Formal verification by Isabelle herself

Why Formal Verification

Why Formal Verification

- Blanchette has finetuned the encodings making them as clutter-free as possible

Why Formal Verification

- Blanchette has finetuned the encodings making them as clutter-free as possible
  He can sleep better: they are still sound!

Why Formal Verification

- Blanchette has finetuned the encodings making them as clutter-free as possible
  He can sleep better: they are still sound!
- General-purpose formalization of MFOL results: completeness, clausification, Löw-Skolem

Why Formal Verification

- Blanchette has finetuned the encodings making them as clutter-free as possible
  He can sleep better: they are still sound!
- General-purpose formalization of MFOL results: completeness, clausification, Löw-Skolem
- Test/illustrate new Isabelle features

Why Formal Verification

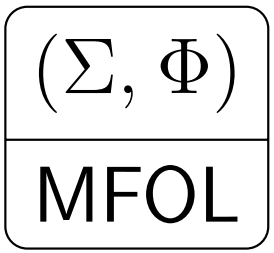
- Blanchette has finetuned the encodings making them as clutter-free as possible
  He can sleep better: they are still sound!
- General-purpose formalization of MFOL results: completeness, clausification, Löw-Skolem
- Test/illustrate new Isabelle features
  - Theory of syntax with bindings

Why Formal Verification

- Blanchette has finetuned the encodings making them as clutter-free as possible
  He can sleep better: they are still sound!
- General-purpose formalization of MFOL results: completeness, clausification, Löw-Skolem
- Test/illustrate new Isabelle features
  - Theory of syntax with bindings
  - Codatatypes

Encodings

Σ: MFOL signature
Φ: set of Σ-formulas



Encodings

$\Sigma^{\#}$: Σ with sorts erased and func/pred added

$$\Phi^{\#} = \mathsf{Ax}_{\Phi} \cup \{\varphi^{\#} \mid \varphi \in \Phi\}$$

(Σ, Φ)
MFOL

$(\Sigma^{\#}, \Phi^{\#})$
UFOL

Encodings

Soundness

$$M \models \Phi \Longrightarrow M^{\#} \models \Phi^{\#}$$

Encodings

Completeness

$$N^{\$} \models \Phi \Longleftarrow N \models \Phi^{\#}$$

Encodings

Example

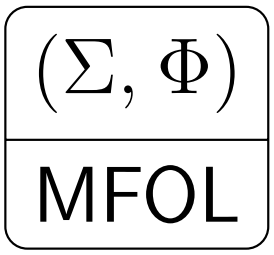
Encodings

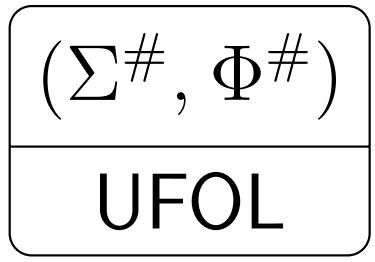
Example

$\Sigma : (\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$

Encodings

Example





Σ : $(\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$
Φ : $\mathsf{nil} \neq \mathsf{cons}(x, xs)$

Encodings

Example unary guard predicates for each erased sort

(Σ, Φ)
MFOL

$(\Sigma^{\#}, \Phi^{\#})$
UFOL

Σ : $(\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$
Φ : $\mathsf{nil} \neq \mathsf{cons}(x, xs)$
$\Sigma^{\#}$: $(\{\mathsf{nil}, \mathsf{cons}\}, \{g_{\mathsf{elem}}, g_{\mathsf{list}}\})$

Encodings

Example unary guard predicates for each erased sort

(Σ, Φ)
MFOL

$(\Sigma^{\#}, \Phi^{\#})$
UFOL

Σ : $(\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$
Φ : $\mathsf{nil} \neq \mathsf{cons}(x, xs)$
$\Sigma^{\#}$: $(\{\mathsf{nil}, \mathsf{cons}\}, \{g_{\mathsf{elem}}, g_{\mathsf{list}}\})$
Ax_{Φ} : $g_{\mathsf{list}}(\mathsf{nil}), \quad g_{\mathsf{list}}(\mathsf{cons}(x, xs))$

Encodings

Example unary guard predicates for each erased sort

(Σ, Φ)
MFOL

$(\Sigma^{\#}, \Phi^{\#})$
UFOL

Σ : $(\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$
Φ : $\mathsf{nil} \neq \mathsf{cons}(x, xs)$
$\Sigma^{\#}$: $(\{\mathsf{nil}, \mathsf{cons}\}, \{g_{\mathsf{elem}}, g_{\mathsf{list}}\})$
Ax_{Φ} : $g_{\mathsf{list}}(\mathsf{nil}), \quad g_{\mathsf{list}}(\mathsf{cons}(x, xs))$
$\Phi^{\#}$: Ax_{Φ} together with
$(\mathsf{nil} \neq \mathsf{cons}(x, xs))^{\#}$

Encodings

Example unary guard predicates for each erased sort

(Σ, Φ)
MFOL

$(\Sigma^{\#}, \Phi^{\#})$
UFOL

Σ : $(\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$
Φ : $\mathsf{nil} \neq \mathsf{cons}(x, xs)$
$\Sigma^{\#}$: $(\{\mathsf{nil}, \mathsf{cons}\}, \{g_{\mathsf{elem}}, g_{\mathsf{list}}\})$
Ax_{Φ} : $g_{\mathsf{list}}(\mathsf{nil}), \quad g_{\mathsf{list}}(\mathsf{cons}(x, xs))$
$\Phi^{\#}$: Ax_{Φ} together with
$g_{\mathsf{elem}}(x) \wedge g_{\mathsf{list}}(xs) \Longrightarrow \mathsf{nil} \neq \mathsf{cons}(x, xs)$

Encodings

Example unary guard predicates for each erased sort

(Σ, Φ)
MFOL

$(\Sigma^{\#}, \Phi^{\#})$
UFOL

Σ : $(\{\mathsf{elem}, \mathsf{list}\}, \{\mathsf{nil}, \mathsf{cons}\}, \emptyset)$

Φ : $\mathsf{nil} \neq \mathsf{cons}(x, xs)$

$\Sigma^{\#}$: $(\{\mathsf{nil}, \mathsf{cons}\}, \{g_{\mathsf{elem}}, g_{\mathsf{list}}\})$

Ax_{Φ} : $g_{\mathsf{list}}(\mathsf{nil}), g_{\mathsf{list}}(\mathsf{cons}(x, xs))$

$\Phi^{\#}$: Ax_{Φ} together with

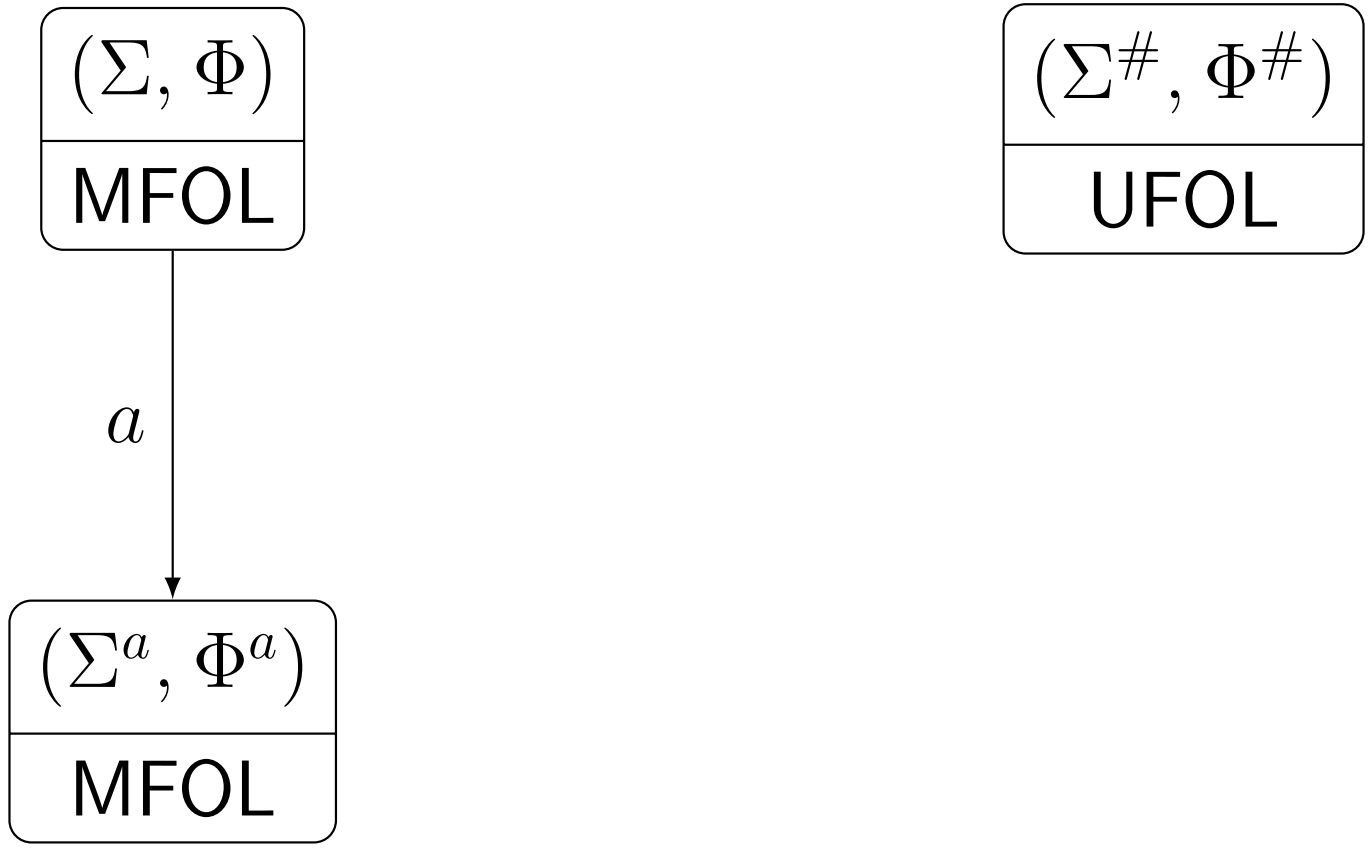
$g_{\mathsf{elem}}(x) \wedge g_{\mathsf{list}}(xs) \implies nil \neq cons(x, xs)$

Encodings

Methodology:

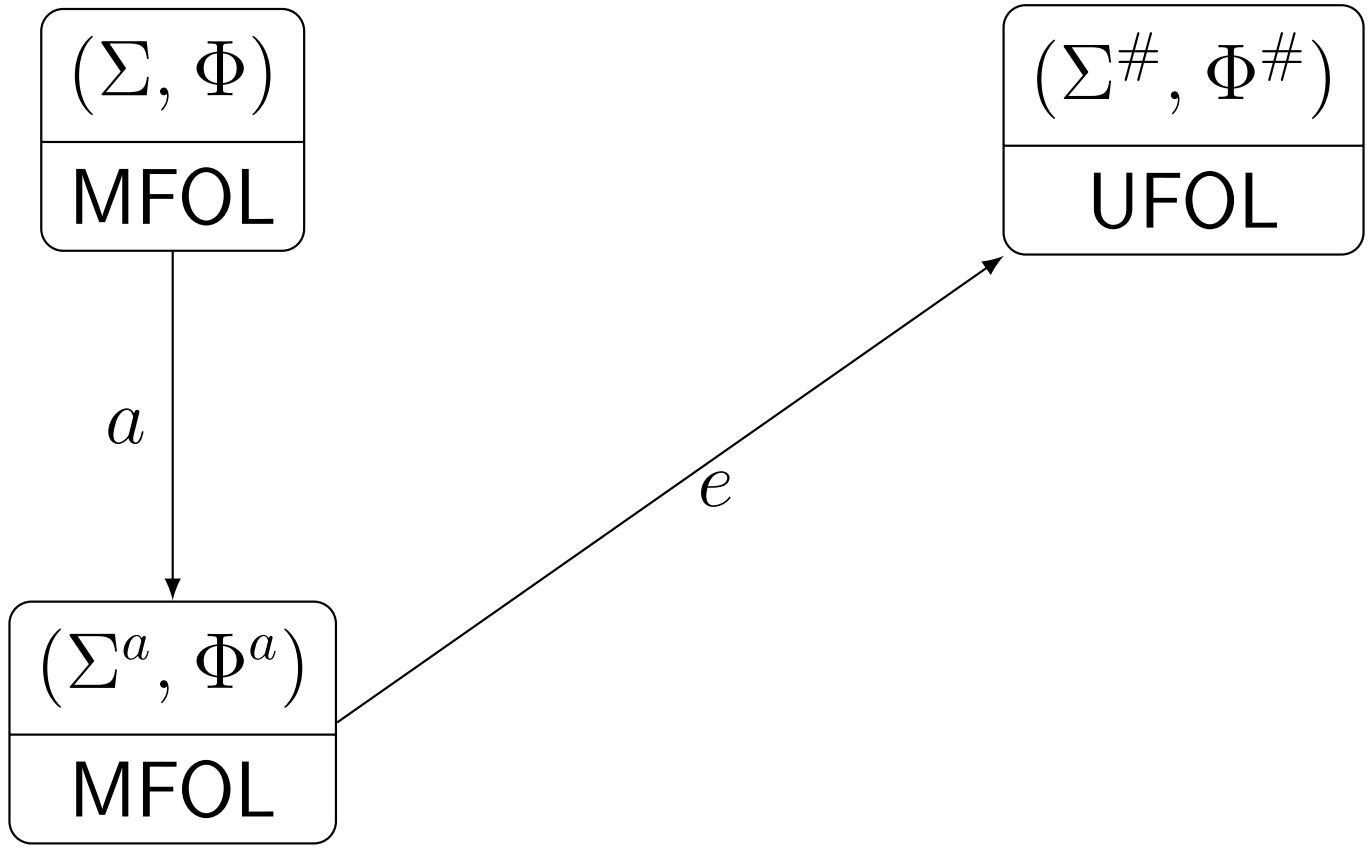
Encodings

Methodology: add decorations first,



Encodings

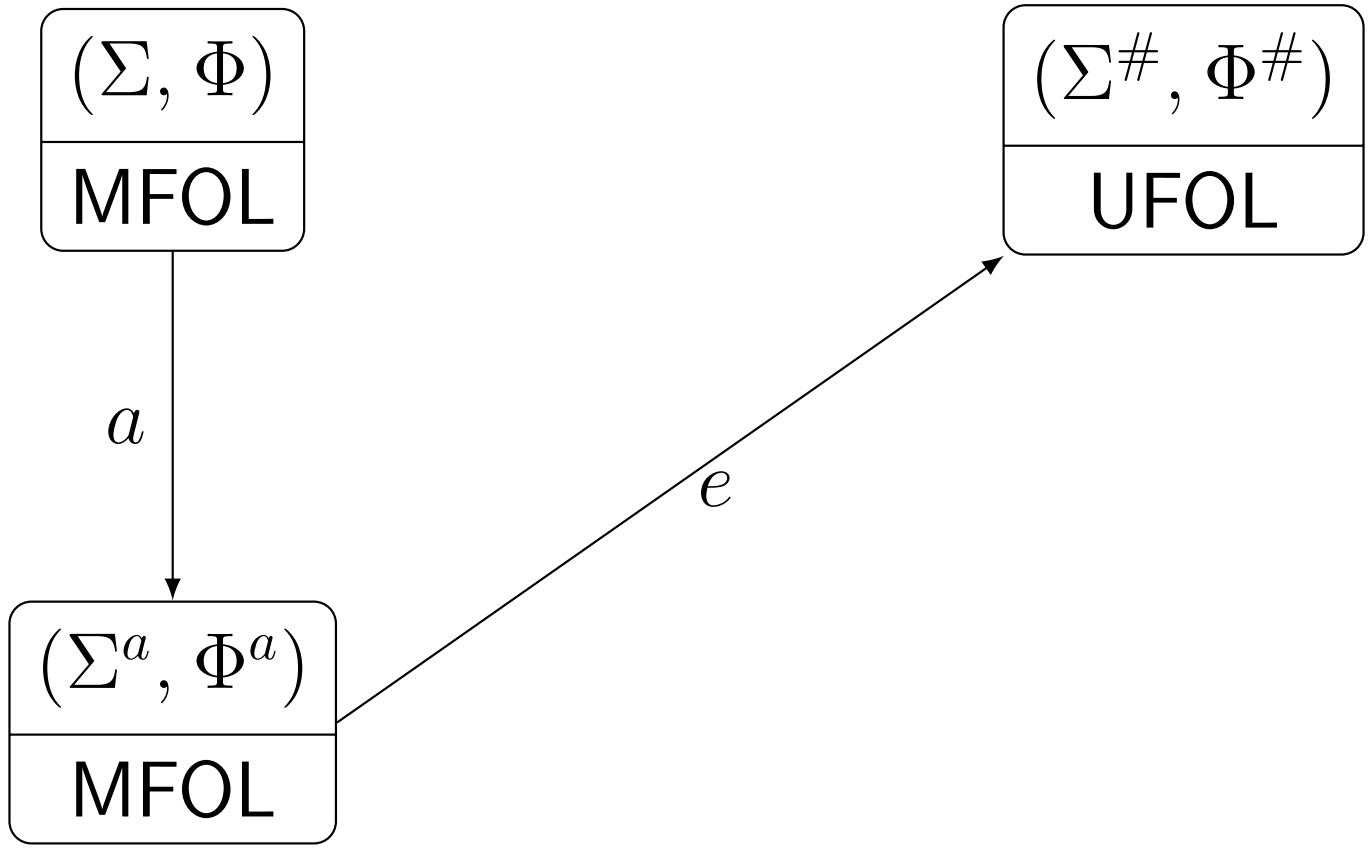
Methodology: add decorations first, erase second



Encodings

Methodology: add decorations first, erase second

$$\# = e \circ a$$

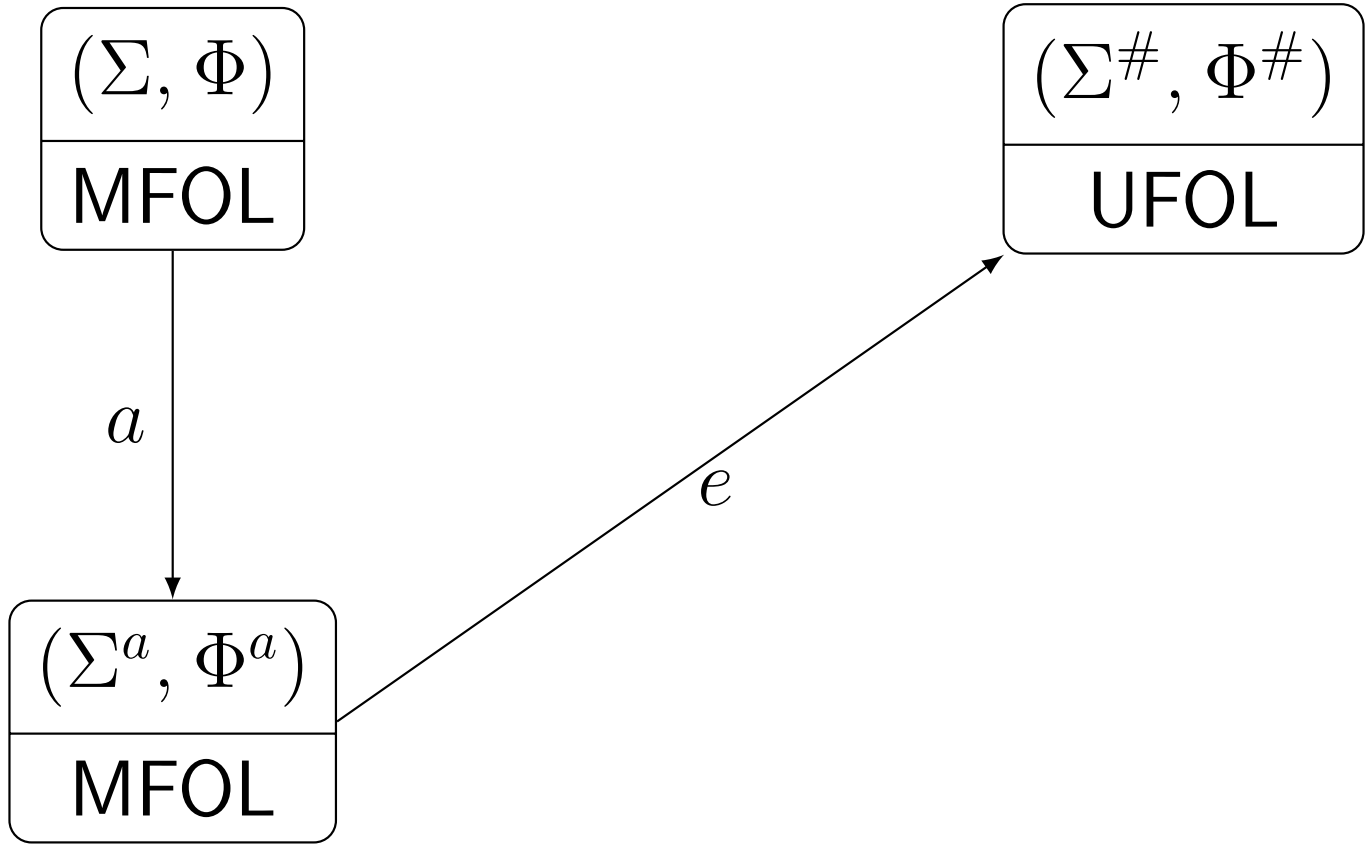


Encodings

Methodology: add decorations first, erase second

$$\# = e \circ a$$

a needs to add enough decorations to make e sound

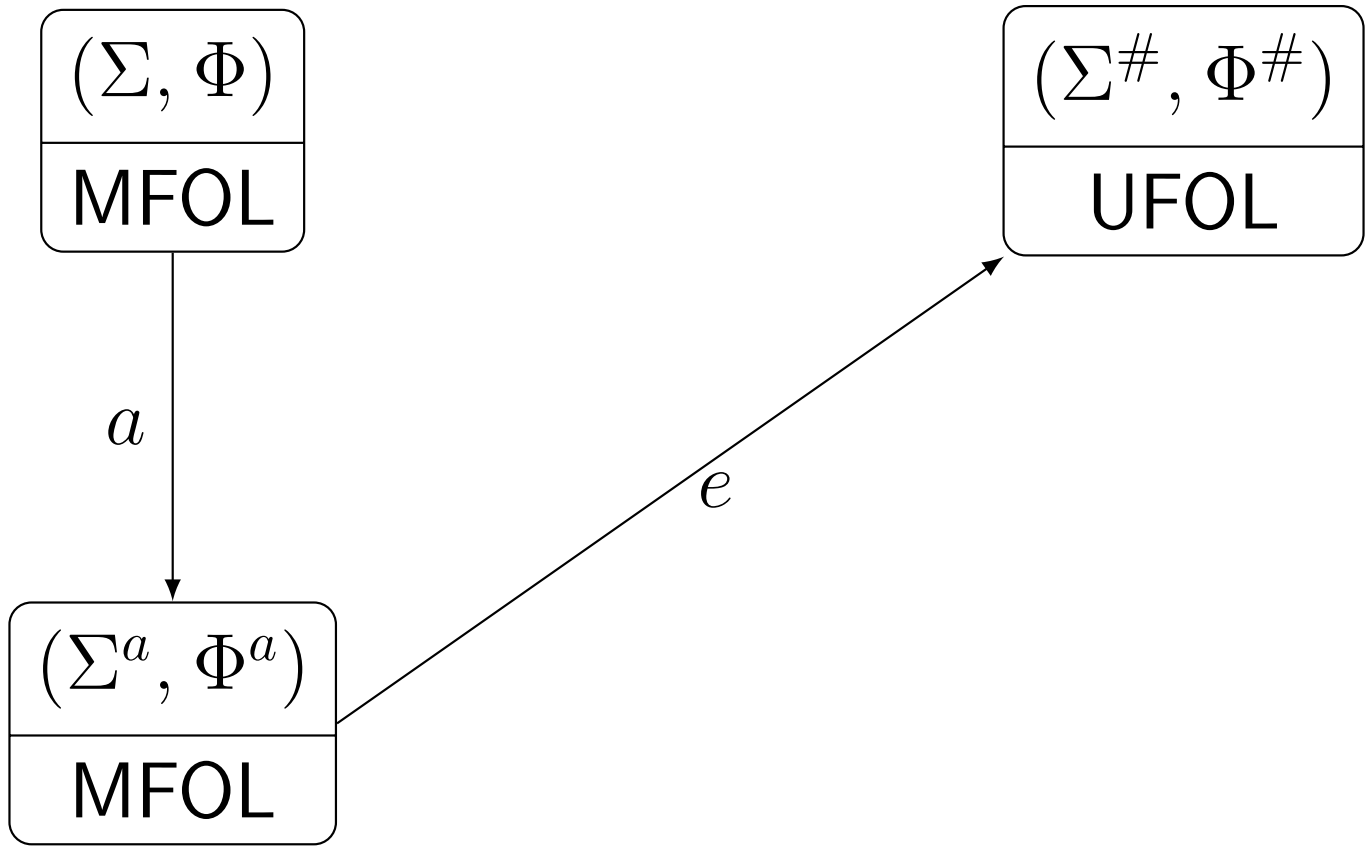


Encodings

Methodology: add decorations first, erase second

$$\# = e \circ a$$

Sufficient: make Φ^a monotonic



Monotonicity

- Claessen et al., CADE 2011:
  Sort It Out with Monotonicity

Monotonicity

- Claessen et al., CADE 2011:
  Sort It Out with Monotonicity
- Φ monotonic: If Φ has a model, then it also has a larger model with all type interpretations of equal size

Monotonicity

- Claessen et al., CADE 2011:
  Sort It Out with Monotonicity
- Φ monotonic: If Φ has a model, then it also has a larger model with all type interpretations of equal size
  $\mapsto$ types in Φ can be safely erased

Monotonicity

- Claessen et al., CADE 2011:
  Sort It Out with Monotonicity
- Φ monotonic: If Φ has a model, then it also has a larger model with all type interpretations of equal size
  $\mapsto$ types in Φ can be safely erased
- nil $\neq$ cons(x, xs) already monotonic

Monotonicity

- Claessen et al., CADE 2011:
  Sort It Out with Monotonicity
- Φ monotonic: If Φ has a model, then it also has a larger model with all type interpretations of equal size
  $\mapsto$ types in Φ can be safely erased
- nil $\neq$ cons(x, xs) already monotonic
- Syntactic criteria

Our Formalization

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Locales: Fix types, constants and assumptions

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Locales: Fix types, constants and assumptions
Problems = Locales

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Locales: Fix types, constants and assumptions
Problems = Locales Encodings = Morphisms

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Locales: Fix types, constants and assumptions

Problems = Locales Encodings = Morphisms

MFOL_Prob $<_a$ MFOL_Mono_Prob $<_e$ UFOL_Prob

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Can focus on at most countable models

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Can focus on at most countable models
Use a fixed countable HOL type

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Can focus on at most countable models
Use a fixed countable HOL type
Monotonicity analysis becomes manageable

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

- MFOL and UFOL
  - bureaucracy
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

New Isabelle theory of syntax with bindings

- MFOL and UFOL
  - eaucracy
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

New Isabelle theory of syntax with bindings
Recursors that interact well with substitution

- MFOL and UFOL
  - acy
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

New Isabelle theory of syntax with bindings
Recursors that interact well with substitution
Optimized for semantic interpretations of syntax

- MFOL and UFOL
  -
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Formalized tableau-based proof a la Beth/Hintikka

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Formalized tableau-based proof a la Beth/Hintikka
Infinite proof trees captured as a codatatype

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Our Formalization

Formalized tableau-based proof a la Beth/Hintikka
Infinite proof trees captured as a codatatype
Used Isabelle's codatatype package

- MFOL and UFOL
  - basics: formulas, models, satisfaction, substitution
  - classic results: completeness, compactness, Löwenheim-Skolem, clausification
- Type encodings from Blanchette et al.
- Monotonicity criteria from Claessen et al.
- Soundness and completeness

Conclusions

Conclusions

- Have put Isabelle to work in order to verify its own communication with ATPs

Conclusions

- Have put Isabelle to work in order to verify its own communication with ATPs
- Based on this, Isabelle might learn (how) to trust the ATPs

Mechanizing the Metatheory of Sledgehammer

Jasmin Blanchette and Andrei Popescu

Fakultät für Informatik
Technische Universität München