Translating proofs from an impredicative type system to a predicative one

Thiago Felicissimo, Frédéric Blanqui, Ashish Kumar Barnawal

Kiryu

November 22, 2022

The problem of proof interoperability

Mechanized proofs can be checked automatically, reused in other proofs

The problem of proof interoperability

Mechanized proofs can be checked automatically, reused in other proofs

Problem Each proof assistant has its own isolated ecosystem

Proofs cannot be reused across different proof assistants

The problem of proof interoperability

Mechanized proofs can be checked automatically, reused in other proofs

Problem Each proof assistant has its own isolated ecosystem

Proofs cannot be reused across different proof assistants

Naive approach Hack the implementation of A to produce proofs in B

But changing codebase of A can break translation

What if proof assistant A' implements same logic as A? Redo everything...

Logical Frameworks for sharing proofs & Dedukti

Solution Define logics in a common system – in a *logical framework*

Proof transformations can be defined inside the system

Transformations do not depend on implementation, but on the logics

Logical Frameworks for sharing proofs & Dedukti

Solution Define logics in a common system – in a *logical framework*

Proof transformations can be defined inside the system

Transformations do not depend on implementation, but on the logics

Dedukti A logical framework for sharing proofs

Sufficiently expressive to define logics of common proof assistants

Used by Thiré to export Fermat's Little Theorem to Coq, HOL, PVS, Lean, etc

Used by Géran to export part of Euclid's Elements to Matita, HOL, Lean, etc

(Im)Predicativity

Impredicative logics allow quantification over arbitrary entities

$$\forall P. P \Rightarrow P$$

(Im)Predicativity

Impredicative logics allow quantification over arbitrary entities

$$\forall P.P \Rightarrow P$$

Challenge Share proofs coming from impredicative proof assistants (Coq, Matita, HOL, etc) with predicative proof assistants (Agda)

(Im)Predicativity

Impredicative logics allow quantification over arbitrary entities

$$\forall P.P \Rightarrow P$$

Challenge Share proofs coming from impredicative proof assistants (Coq, Matita, HOL, etc) with predicative proof assistants (Agda)

Fact Proofs using impredicativity in an essential way cannot be shared

But do most proofs really use impredicativity?

If not, how to translate them?

Our contribution

We propose a transformation for sharing proofs with predicative systems

Non essential impredicativity is replaced by universe-polymorphism

Leads us to study unification over universe levels

Our contribution

We propose a transformation for sharing proofs with predicative systems

Non essential impredicativity is replaced by universe-polymorphism

Leads us to study unification over universe levels

Transformation implemented in the tool PREDICATIVIZE

Allowed to translate MATITA's arithmetic library to (safe) AGDA

First proofs of Fermat's Little Theorem and Bertrand's Postulate in AGDA

Outline

An informal look at proof predicativization

A Universe-Polymorphic Predicative System

The translation

- Solving constraints on universe levels

Predicativize & translating Matita's library to Agda

Dedukti

A λ-calculus with dependent types and an extensible equality

Dedukti

A λ-calculus with dependent types and an extensible equality

Rewrite rules allow to express computation

$$n + (succ\ m) \hookrightarrow succ\ (n + m) \in \mathscr{R}$$

And non-computational equalities can be expressed using *rewrite modulo*

$$n + m \simeq m + n \in \mathscr{E}$$

Dedukti

A λ-calculus with dependent types and an extensible equality

Rewrite rules allow to express computation

$$n + (succ\ m) \hookrightarrow succ\ (n + m) \in \mathscr{R}$$

And non-computational equalities can be expressed using *rewrite modulo*

$$n + m \simeq m + n \in \mathscr{E}$$

Dedukti Theory A signature Σ, rewrite rules $\mathscr{R}$, equations $\mathscr{E}$

Allows to define an object logic inside Dedukti

An useful example of Dedukti theory: Pure Type Systems

Universes specified by a set $\mathcal{S}$

$$U_s : \textbf{Type} \qquad \text{for } s \in \mathcal{S}$$

$$El_s : U_s \to \textbf{Type} \qquad \text{for } s \in \mathcal{S}$$

An useful example of Dedukti theory: Pure Type Systems

Universes specified by a set $\mathcal{S}$

$$U_s : \textbf{Type} \qquad \text{for } s \in \mathcal{S}$$

$$El_s : U_s \to \textbf{Type} \qquad \text{for } s \in \mathcal{S}$$

and a binary relation $\mathcal{A} \subseteq \mathcal{S}^2$

$$u_{s_1} : U_{s_2} \qquad \text{for } (s_1, s_2) \in \mathcal{A}$$

$$El_{s_2}\ u_{s_1} \hookrightarrow U_{s_1} \qquad \text{for } (s_1, s_2) \in \mathcal{A}$$

An useful example of Dedukti theory: Pure Type Systems

Universes specified by a set $\mathcal{S}$

$$U_s : \textbf{Type} \qquad \text{for } s \in \mathcal{S}$$

$$El_s : U_s \to \textbf{Type} \qquad \text{for } s \in \mathcal{S}$$

and a binary relation $\mathcal{A} \subseteq \mathcal{S}^2$

$$u_{s_1} : U_{s_2} \qquad \text{for } (s_1, s_2) \in \mathcal{A}$$

$$El_{s_2}\ u_{s_1} \hookrightarrow U_{s_1} \qquad \text{for } (s_1, s_2) \in \mathcal{A}$$

Dependent functions specified by a trinary relation $\mathcal{R} \subseteq \mathcal{S}^3$

$$\pi_{s_1,s_2} : \Pi(A : U_{s_1}).(El_{s_1}\ A \to U_{s_2}) \to U_{s_3} \qquad \text{for } (s_1, s_2, s_3) \in \mathcal{R}$$

$$El_{s_3}\ (\pi_{s_1,s_2}\ A\ B) \hookrightarrow \Pi x : El_{s_1}\ A.El_{s_2}\ (B\ x) \qquad \text{for } (s_1, s_2, s_3) \in \mathcal{R}$$

We write $\pi_{s_1,s_2}\ A\ (\lambda x.B)$ as $\pi_{s_1,s_2} x : A.B$ or $A \rightsquigarrow_{s_1,s_2} B$ when $x \notin B$

An informal look at proof predicativization

The problem of predicativization

Consider the PTSs **I** (impredicative) and **P** (predicative) specified by

$$\mathcal{S}_{\boldsymbol{I}} = \{*, \square\} \qquad \mathcal{S}_{\boldsymbol{P}} = \mathbb{N}$$

$$\mathcal{A}_{\boldsymbol{I}} = \{(*, \square)\} \qquad \mathcal{A}_{\boldsymbol{P}} = \{(n, n+1) \mid n \in \mathbb{N}\}$$

$$\mathcal{R}_{\boldsymbol{I}} = \{(*, *, *), (\square, *, *), (\square, \square, \square)\} \qquad \mathcal{R}_{\boldsymbol{P}} = \{(n, m, max\{n, m\}) \mid n, m \in \mathbb{N}\}$$

The problem of predicativization

Consider the PTSs **I** (impredicative) and **P** (predicative) specified by

$$\mathcal{S}_{\boldsymbol{I}} = \{*, \square\} \qquad \mathcal{S}_{\boldsymbol{P}} = \mathbb{N}$$

$$\mathcal{A}_{\boldsymbol{I}} = \{(*, \square)\} \qquad \mathcal{A}_{\boldsymbol{P}} = \{(n, n+1) \mid n \in \mathbb{N}\}$$

$$\mathcal{R}_{\boldsymbol{I}} = \{(*, *, *), (\square, *, *), (\square, \square, \square)\} \qquad \mathcal{R}_{\boldsymbol{P}} = \{(n, m, max\{n, m\}) \mid n, m \in \mathbb{N}\}$$

Predicativization Given a signature Δ containing declarations $c : A$ and definitions $c : A := t$ well-formed in **I**, translate it into a signature Δ' well-formed in **P**

The problem of predicativization: first try

Mapping $*$ to 0 and $\square$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

The problem of predicativization: first try

Mapping $*$ to 0 and $\Box$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_* \; (\pi_{\Box,*} \; P : u_*.P \rightsquigarrow_{*,*} P) := \lambda P : U_*.\lambda p : El_* \; P.p$$

The problem of predicativization: first try

Mapping $*$ to 0 and $\square$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) := \lambda P : U_0.\lambda p : El_0\ P.p$$

The problem of predicativization: first try

Mapping $*$ to 0 and $\square$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) := \lambda P : U_0.\lambda p : El_0\ P.p \quad \text{OK}$$

The problem of predicativization: first try

Mapping $*$ to 0 and $\square$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) := \lambda P : U_0.\lambda p : El_0\ P.p \quad \text{OK}$$

But...

$$thm_3 : El_*\ ((\pi_{\square,*}\ P : u_*.P \rightsquigarrow_{*,*} P) \rightsquigarrow_{*,*} \pi_{\square,*}\ P : u_*.P \rightsquigarrow_{*,*} P)$$
$$:= thm_1\ (\pi_{\square,*}\ P : u_*.P \rightsquigarrow_{*,*} P)$$

The problem of predicativization: first try

Mapping $*$ to 0 and $\square$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) := \lambda P : U_0.\lambda p : El_0\ P.p \quad \text{OK}$$

But...

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$
$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

The problem of predicativization: first try

Mapping $*$ to 0 and $\Box$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) := \lambda P : U_0.\lambda p : El_0\ P.p \quad \text{OK}$$

But...

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$
$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{KO}$$

The problem of predicativization: first try

Mapping $*$ to 0 and $\Box$ to 1 in U and u, recomputing annotations of El, π, $\rightsquigarrow$

$$thm_1 : El_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) := \lambda P : U_0.\lambda p : El_0\ P.p \quad \text{OK}$$

But...

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$
$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{KO}$$

thm_1 expects type in U_0, but $\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P$ lives in U_1

Impredicativity as **typical ambiguity**

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_* \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) := \lambda P : U_*.\lambda p : El_* \ P.p$$

$$thm_3 : El_* \ ((\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) \rightsquigarrow_{*,*} \pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P)$$

$$:= thm_1 \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P)$$

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_{l_1}\ (\pi_{l_2,l_3}\ P : u_{l_4}.P \rightsquigarrow_{l_5,l_6} P) := \lambda P : U_{l_7}.\lambda p : El_{l_8}\ P.p$$

$$thm_3 : El_{l_9}\ ((\pi_{l_{10},l_{11}}\ P : u_{l_{12}}.P \rightsquigarrow_{l_{13},l_{14}} P) \rightsquigarrow_{l_{15},l_{16}} \pi_{l_{17},l_{18}}\ P : \ u_{l_{19}}.P \rightsquigarrow_{l_{20},l_{21}} P)$$

$$:= thm_1\ (\pi_{l_{22},l_{23}}\ P : u_{l_{24}}.P \rightsquigarrow_{l_{25},l_{26}} P)$$

The problem of predicativization: second try

For each occurrence of $*$ and $\Box$, compute some adequate predicative universe

$$thm_1 : El_2\ (\pi_{2,1}\ P : u_1.P \rightsquigarrow_{1,1} P)) := \lambda P : U_1.\lambda p : El_1\ P.p$$

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_2\ (\pi_{2,1}\ P : u_1.P \rightsquigarrow_{1,1} P)) := \lambda P : U_1.\lambda p : El_1\ P.p$$

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{OK}$$

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_2\ (\pi_{2,1}\ P : u_1.P \rightsquigarrow_{1,1} P)) := \lambda P : U_1.\lambda p : El_1\ P.p$$

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{OK}$$

But...

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_2\ (\pi_{2,1}\ P : u_1.P \rightsquigarrow_{1,1} P)) := \lambda P : U_1.\lambda p : El_1\ P.p$$

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$
$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{OK}$$

But...

$$thm_1 : El_*\ (\pi_{\square,*}\ P : u_*.P \rightsquigarrow_{*,*} P) := \lambda P : U_*.\lambda p : El_*\ P.p$$

$$thm_2 : El_*\ (\pi_{\square,*}\ P : u_*.P \rightsquigarrow_{*,*} P) := thm_1\ (\pi_{\square,*}\ P : u_*.P \rightsquigarrow_{*,*} P)\ thm_1$$

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_2\ (\pi_{2,1}\ P : u_1.P \rightsquigarrow_{1,1} P)) := \lambda P : U_1.\lambda p : El_1\ P.p$$

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{OK}$$

But...

$$thm_1 : El_{l_1}\ (\pi_{l_2,l_3}\ P : u_{l_4}.P \rightsquigarrow_{l_5,l_6} P) := \lambda P : U_{l_7}.\lambda p : El_{l_8}\ P.p$$

$$thm_2 : El_{l_9}\ (\pi_{l_{10},l_{11}}\ P : u_{l_{12}}.P \rightsquigarrow_{l_{13},l_{14}} P) := thm_1\ (\pi_{l_{15},l_{16}}\ P : u_{l_{17}}.P \rightsquigarrow_{l_{18},l_{19}} P)\ thm_1$$

The problem of predicativization: second try

For each occurrence of $*$ and $\square$, compute some adequate predicative universe

$$thm_1 : El_2\ (\pi_{2,1}\ P : u_1.P \rightsquigarrow_{1,1} P)) := \lambda P : U_1.\lambda p : El_1\ P.p$$

$$thm_3 : El_1\ ((\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \rightsquigarrow_{1,1} \pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P)$$

$$:= thm_1\ (\pi_{1,0}\ P : u_0.P \rightsquigarrow_{0,0} P) \quad \text{OK}$$

But...

$$thm_1 : El_{l_1}\ (\pi_{l_2,l_3}\ P : u_{l_4}.P \rightsquigarrow_{l_5,l_6} P) := \lambda P : U_{l_7}.\lambda p : El_{l_8}\ P.p \quad \text{KO}$$

$$thm_2 : El_{l_9}\ (\pi_{l_{10},l_{11}}\ P : u_{l_{12}}.P \rightsquigarrow_{l_{13},l_{14}} P) := thm_1\ (\pi_{l_{15},l_{16}}\ P : u_{l_{17}}.P \rightsquigarrow_{l_{18},l_{19}} P)\ thm_1$$

Unsolvable constraints thm_1 needs to be at two universes at same time

We compensate this by using **universe-polymorphism**

A Universe-Polymorphic Predicative System

A Universe-Polymorphic Predicative System (UPP)

Universe levels internalized inside the framework

A Universe-Polymorphic Predicative System (UPP)

Universe levels internalized inside the framework

$$\mathit{Level} : \mathbf{Type} \qquad \mathtt{s} : \mathit{Level} \to \mathit{Level}$$

$$\mathtt{z} : \mathit{Level} \qquad \sqcup : \mathit{Level} \to \mathit{Level} \to \mathit{Level} \quad \text{(written infix)}$$

A Universe-Polymorphic Predicative System (UPP)

Universe levels internalized inside the framework

$Level$: **Type**

$\mathtt{z} : Level$

$\mathtt{s} : Level \to Level$

$\sqcup : Level \to Level \to Level$ (written infix)

U_s : **Type**

$El_s : U_s \to$ **Type**

$u_{s_1} : U_{s_2}$

$\pi_{s_1,s_2} : \Pi(A : U_{s_1}).(El_{s_1}\ A \to U_{s_2}) \to U_{s_3}$

$El_{s_2}\ u_{s_1} \hookrightarrow U_{s_1}$

$El_{s_3}\ (\pi_{s_1,s_2}\ A\ B) \hookrightarrow \Pi x : El_{s_1}\ A.El_{s_2}\ (B\ x)$

A Universe-Polymorphic Predicative System (UPP)

Universe levels internalized inside the framework

$$\mathit{Level} : \textbf{Type} \qquad \mathtt{s} : \mathit{Level} \to \mathit{Level}$$

$$\mathtt{z} : \mathit{Level} \qquad \sqcup : \mathit{Level} \to \mathit{Level} \to \mathit{Level} \quad \text{(written infix)}$$

$$U : \mathit{Level} \to \textbf{Type} \qquad \pi : \Pi(i_A\ i_B : \mathit{Level})\ (A : U\ i_A).(El\ i_A\ A \to U\ i_B) \to U\ (i_A \sqcup i_B)$$

$$El : \Pi(i : \mathit{Level}).U\ i \to \textbf{Type} \qquad El\ i'\ (u\ i) \hookrightarrow U\ i$$

$$u : \Pi(i : \mathit{Level}).U\ (\mathtt{s}\ i) \qquad El\ i'\ (\pi\ i_A\ i_B\ A\ B) \hookrightarrow \Pi x : El\ i_A\ A.El\ i_B\ (B\ x)$$

A Universe-Polymorphic Predicative System (UPP)

Universe levels internalized inside the framework

$\mathit{Level} : \mathbf{Type}$	$\mathtt{s} : \mathit{Level} \to \mathit{Level}$
$\mathtt{z} : \mathit{Level}$	$\sqcup : \mathit{Level} \to \mathit{Level} \to \mathit{Level}$ (written infix)
$U : \mathit{Level} \to \mathbf{Type}$	$\pi : \Pi(i_A\ i_B : \mathit{Level})\ (A : U\ i_A).(\mathit{El}\ i_A\ A \to U\ i_B) \to U\ (i_A \sqcup i_B)$
$\mathit{El} : \Pi(i : \mathit{Level}).U\ i \to \mathbf{Type}$	$\mathit{El}\ i'\ (u\ i) \hookrightarrow U\ i$
$u : \Pi(i : \mathit{Level}).U\ (\mathtt{s}\ i)$	$\mathit{El}\ i'\ (\pi\ i_A\ i_B\ A\ B) \hookrightarrow \Pi x : \mathit{El}\ i_A\ A.\mathit{El}\ i_B\ (B\ x)$

Universe-polymorphism as quantification over levels

$$\lambda i.\lambda A : U_i.\lambda a : \mathit{El}_i\ A.a \quad : \quad \Pi(i : \mathit{Level}).\mathit{El}_{(\mathtt{s}\ i)}\ (\pi_{(\mathtt{s}\ i),i}\ A : u_i.A \rightsquigarrow_{i,i} A)$$

Level equality

Levels are not purely syntactic entities! Their equality is defined by:

$$i_1 \sqcup (i_2 \sqcup i_3) \simeq (i_1 \sqcup i_2) \sqcup i_3 \qquad \text{(Associativity)}$$

$$i_1 \sqcup i_2 \simeq i_2 \sqcup i_1 \qquad \text{(Commutativity)}$$

$$i \sqcup \mathbf{z} \simeq i \qquad \text{(Unit)}$$

$$i \sqcup i \simeq i \qquad \text{(Idempotency)}$$

$$\mathbf{s}\ (i_1 \sqcup i_2) \simeq \mathbf{s}\ i_1 \sqcup \mathbf{s}\ i_2 \qquad \text{(Distributivity)}$$

$$i \sqcup \mathbf{s}\ i \simeq \mathbf{s}\ i \qquad \text{(Subsumption)}$$

Level equality

Levels are not purely syntactic entities! Their equality is defined by:

$$i_1 \sqcup (i_2 \sqcup i_3) \simeq (i_1 \sqcup i_2) \sqcup i_3 \qquad \text{(Associativity)}$$

$$i_1 \sqcup i_2 \simeq i_2 \sqcup i_1 \qquad \text{(Commutativity)}$$

$$i \sqcup \mathbf{z} \simeq i \qquad \text{(Unit)}$$

$$i \sqcup i \simeq i \qquad \text{(Idempotency)}$$

$$\mathbf{s}\ (i_1 \sqcup i_2) \simeq \mathbf{s}\ i_1 \sqcup \mathbf{s}\ i_2 \qquad \text{(Distributivity)}$$

$$i \sqcup \mathbf{s}\ i \simeq \mathbf{s}\ i \qquad \text{(Subsumption)}$$

Given a valuation $\sigma : \mathcal{I} \to \mathbb{N}$, let $[\![l]\!]_\sigma$ be the interpretation of a level l in $\mathbb{N}$

Justifying property $l \simeq l'$ iff $[\![l]\!]_\sigma = [\![l']\!]_\sigma$ for all σ

The translation

The translation

Given a signature Δ in **I**, we translate each entry by dependency order to **UPP**

The translation

Given a signature Δ in **I**, we translate each entry by dependency order to **UPP**

We illustrate the translation with the signature

$$\Delta_I = thm_1 : El_* \; (\pi_{\square,*} \; P : u_*.P \rightsquigarrow_{*,*} P) := \lambda P : U_*.\lambda p : El_* \; P.p \; ,$$
$$thm_2 : El_* \; (\pi_{\square,*} \; P : u_*.P \rightsquigarrow_{*,*} P) := thm_1 \; (\pi_{\square,*} \; P : u_*.P \rightsquigarrow_{*,*} P) \; thm_1$$

The translation

Given a signature Δ in **I**, we translate each entry by dependency order to **UPP**

We illustrate the translation with the signature

$$\Delta_I = thm_1 : El_* \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) := \lambda P : U_*.\lambda p : El_* \ P.p \ ,$$
$$thm_2 : El_* \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) := thm_1 \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) \ thm_1$$

Suppose its first entry has already been translated, giving the signature

$$\Delta_{thm_1} = thm_1 : \Pi(i : Level).El_{(\mathtt{s}\ i)} \ (\pi_{(\mathtt{s}\ i),i} \ P : u_i.P \rightsquigarrow_{i,i} P) := \lambda i.\lambda P : U_i.\lambda p : El_i \ P.p$$

Let us translate thm_2

The translation

First step Insert fresh level metavariables

$\textsc{InsertMetas}(El_s) = El_i$

$\textsc{InsertMetas}(U_s) = U_i$

$\textsc{InsertMetas}(u_s) = u_i$

$\textsc{InsertMetas}(\pi_{s_1,s_2}) = \pi_{i,j}$

$\textsc{InsertMetas}(c) = c\ i_1 \ldots i_k$ where c expects k level arguments

$\textsc{InsertMetas}(M) = M$ if M is a variable, **Type** or **Kind**

$\textsc{InsertMetas}(\Pi x : A.B) = \Pi x : \textsc{InsertMetas}(A).\textsc{InsertMetas}(B)$

$\textsc{InsertMetas}(\lambda x : A.M) = \lambda x : \textsc{InsertMetas}(A).\textsc{InsertMetas}(M)$

$\textsc{InsertMetas}(MN) = \textsc{InsertMetas}(M)\ \textsc{InsertMetas}(N)$

The translation

First step Insert fresh level metavariables

$$thm_2 : El_* \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) := thm_1 \ (\pi_{\square,*} \ P : u_*.P \rightsquigarrow_{*,*} P) \ thm_1$$

The translation

First step Insert fresh level metavariables

$$thm_2 : El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) := thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$

The translation

First step Insert fresh level metavariables

$$thm_2 : El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) := thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$

Second step Calculate constraints on levels for definition to be valid in **UPP**

$$\frac{c : A := M \in \Sigma_{UPP}, \Delta \text{ or } c : A \in \Sigma_{UPP}, \Delta}{\Sigma_{UPP}, \Delta; \Gamma \vdash c \Rightarrow A \downarrow \emptyset}\ \textsc{Cons} \qquad \frac{x : A \in \Gamma}{\Sigma_{UPP}, \Delta; \Gamma \vdash x \Rightarrow A \downarrow \emptyset}\ \textsc{Var}$$

$$\frac{i \in \mathcal{M}}{\Sigma_{UPP}, \Delta; \Gamma \vdash i \Rightarrow \mathit{Level} \downarrow \emptyset}\ \textsc{Lvl-Var} \qquad \frac{}{\Sigma_{UPP}, \Delta; \Gamma \vdash \mathbf{Type} \Rightarrow \mathbf{Kind} \downarrow \emptyset}\ \textsc{Sort}$$

$$\frac{\Sigma_{UPP}, \Delta; \Gamma \vdash A \Leftarrow \mathbf{Type} \downarrow C_1 \qquad \Sigma_{UPP}, \Delta; \Gamma, x : A \vdash B \Rightarrow_{sort} \mathbf{s} \downarrow C_2}{\Sigma_{UPP}, \Delta; \Gamma \vdash \Pi x : A.B \Rightarrow \mathbf{s} \downarrow C_1 \cup C_2}\ \textsc{Prod}$$

$$\frac{\Sigma_{UPP}, \Delta; \Gamma \vdash A \Leftarrow \mathbf{Type} \downarrow C_1 \qquad \Sigma_{UPP}, \Delta; \Gamma, x : A \vdash M \Rightarrow B \downarrow C_3 \qquad \Sigma_{UPP}, \Delta; \Gamma, x : A \vdash B \Rightarrow_{sort} \mathbf{s} \downarrow C_2}{\Sigma_{UPP}, \Delta; \Gamma \vdash \lambda x : A.M \Rightarrow \Pi x : A.B \downarrow C_1 \cup C_2 \cup C_3}\ \textsc{Abs}$$

$$\frac{\Sigma_{UPP}, \Delta; \Gamma \vdash M \Rightarrow_{\Pi} \Pi x : A.B \downarrow C_1 \qquad \Sigma_{UPP}, \Delta; \Gamma \vdash N \Leftarrow A \downarrow C_2}{\Sigma_{UPP}, \Delta; \Gamma \vdash MN \Rightarrow B\{N/x\} \downarrow C_1 \cup C_2}\ \textsc{App}$$

$$\frac{\Sigma_{UPP}, \Delta; \Gamma \vdash M \Rightarrow A \downarrow C_1 \qquad \mathrm{WHNF}(A) \equiv^{?} \mathrm{WHNF}(B) \downarrow C_2}{\Sigma_{UPP}, \Delta; \Gamma \vdash M \Leftarrow B \downarrow C_1 \cup C_2}\ \textsc{Check}$$

The translation

First step Insert fresh level metavariables

$$thm_2 : El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) := thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$

Second step Calculate constraints on levels for definition to be valid in **UPP**

The translation

First step Insert fresh level metavariables

$$thm_2 : El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) := thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$

Second step Calculate constraints on levels for definition to be valid in **UPP**

$$\Sigma_{UPP}, \Delta_{thm_1}; - \vdash El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) \Rightarrow_{sort} \mathbf{s} \downarrow \{i_2 \sqcup i_3 = i_1, \mathrm{s}\ i_4 = i_2, \ldots\}$$

The translation

First step Insert fresh level metavariables

$$thm_2 : El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) := thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$

Second step Calculate constraints on levels for definition to be valid in **UPP**

$$\Sigma_{UPP}, \Delta_{thm_1}; - \vdash El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) \Rightarrow_{sort} \mathbf{s} \downarrow \{i_2 \sqcup i_3 = i_1, \mathtt{s}\ i_4 = i_2, ...\}$$

$$\Sigma_{UPP}, \Delta_{thm_1}; - \vdash thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$
$$\Leftarrow El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) \downarrow \{i_8 \sqcup i_9 = i_{10}, \mathtt{s}\ i_{10} = i_9, ...\}$$

Let C_{thm_2} be the resulting constraints

The translation

Third step Solve the constraints!

But to allow definition to be used at different levels, we need a general symbolic solution. We thus need *level unification*, which we will see in the next slide

Solution of C_{thm_2} sends all variables to i_4, except i_1, i_2, i_7, i_8, sent to $\mathtt{s}\ i_4$

The translation

Third step Solve the constraints!

But to allow definition to be used at different levels, we need a general symbolic solution. We thus need *level unification*, which we will see in the next slide

Solution of C_{thm_2} sends all variables to i_4, except i_1, i_2, i_7, i_8, sent to $\mathtt{s}\ i_4$

Fourth step Apply substitution and generalize over free level variables

$$thm_2 : El_{i_1}\ (\pi_{i_2,i_3}\ P : u_{i_4}.P \rightsquigarrow_{i_5,i_6} P) :=$$
$$thm_1\ i_7\ (\pi_{i_8,i_9}\ P : u_{i_{10}}.P \rightsquigarrow_{i_{11},i_{12}} P)\ (thm_1\ i_{13})$$

The translation

Third step Solve the constraints!

But to allow definition to be used at different levels, we need a general symbolic solution. We thus need *level unification*, which we will see in the next slide

Solution of C_{thm_2} sends all variables to i_4, except i_1, i_2, i_7, i_8, sent to $\mathtt{s}\ i_4$

Fourth step Apply substitution and generalize over free level variables

$$thm_2 : El_{(\mathtt{s}\ i_4)}\ (\pi_{(\mathtt{s}\ i_4),i_4}\ P : u_{i_4}.P \rightsquigarrow_{i_4,i_4} P) :=$$
$$thm_1\ (\mathtt{s}\ i_4)\ (\pi_{(\mathtt{s}\ i_4),i_4}\ P : u_{i_4}.P \rightsquigarrow_{i_4,i_4} P)\ (thm_1\ i_4)$$

The translation

Third step Solve the constraints!

But to allow definition to be used at different levels, we need a general symbolic solution. We thus need *level unification*, which we will see in the next slide

Solution of C_{thm_2} sends all variables to i_4, except i_1, i_2, i_7, i_8, sent to $\mathtt{s}\ i_4$

Fourth step Apply substitution and generalize over free level variables

$$
\begin{aligned}
thm_2 : \Pi(i_4 : Level).El_{(\mathtt{s}\ i_4)}\ (\pi_{(\mathtt{s}\ i_4),i_4}\ P : u_{i_4}.P \rightsquigarrow_{i_4,i_4} P) := \\
\lambda i_4.thm_1\ (\mathtt{s}\ i_4)\ (\pi_{(\mathtt{s}\ i_4),i_4}\ P : u_{i_4}.P \rightsquigarrow_{i_4,i_4} P)\ (thm_1\ i_4)
\end{aligned}
$$

The translation

Third step Solve the constraints!

But to allow definition to be used at different levels, we need a general symbolic solution. We thus need *level unification*, which we will see in the next slide

Solution of C_{thm_2} sends all variables to i_4, except i_1, i_2, i_7, i_8, sent to $\mathtt{s}\ i_4$

Fourth step Apply substitution and generalize over free level variables

$$
\begin{aligned}
thm_2 : \Pi(i_4 : Level).El_{(\mathtt{s}\ i_4)}\ (\pi_{(\mathtt{s}\ i_4),i_4}\ P : u_{i_4}.P \rightsquigarrow_{i_4,i_4} P) :=\\
\lambda i_4.thm_1\ (\mathtt{s}\ i_4)\ (\pi_{(\mathtt{s}\ i_4),i_4}\ P : u_{i_4}.P \rightsquigarrow_{i_4,i_4} P)\ (thm_1\ i_4)
\end{aligned}
$$

Theorem If all steps succeed, the obtained signature is well-formed in **UPP**

Solving constraints on universe levels

Level unification Given set of constraints C, find substitution θ that solves them

Solving constraints on universe levels

Level unification Given set of constraints C, find substitution θ that solves them

Theorem Not all level unification problems have most general unifiers

Solving constraints on universe levels

Level unification Given set of constraints C, find substitution θ that solves them

Theorem Not all level unification problems have most general unifiers

We propose an incomplete unification algorithm, sufficiently powerful for our needs

The unification algorithm

(**Trivial**)	$\{l = l\} \cup C; \theta \rightsquigarrow C; \theta$	
(**Orient**)	$\{l = l'\} \cup C; \theta \rightsquigarrow \{l' = l\} \cup C; \theta$	if $l' = \mathbf{z}$ or $\mathbf{z} \sqcup i$
(**Eliminate 1**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow \widehat{C\{l/i\}}; \widehat{\theta\{l/i\}}, i \mapsto l$	if $i \notin l$
(**Eliminate 2**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow$	if $\mathbf{s}^m\ i \in l$ with $m = 0$
	let $l' = l\{i'/i\}$ in $\widehat{C\{l'/i\}}; \widehat{\theta\{l'/i\}}, i \mapsto l'$	for some fresh i'
(**Decompose**)	$\{\mathbf{z} = \mathbf{z} \sqcup (\sqcup_{i \in V}\ i)\} \cup C; \theta \rightsquigarrow \{\mathbf{z} \sqcup i = \mathbf{z}\}_{i \in V} \cup C; \theta$	
(**Clash**)	$\{\mathbf{z} = l\} \cup C; \theta \rightsquigarrow \bot$	if $\mathbf{s} \in l$

Three outcomes We get an mgu, there is not solution, or we're stuck

The unification algorithm

Let $C = \{i_1 \sqcup \mathtt{s}\ (\mathtt{s}\ i_2 \sqcup \mathtt{s}\ i_1) = i_2 \sqcup \mathtt{s}\ (\mathtt{s}\ i_1), \quad i_1 \sqcup i_2 \sqcup i_3 = \mathtt{s}\ i_1\}$

$$
\begin{aligned}
i_1 \sqcup \mathtt{s}\ (\mathtt{s}\ i_2 \sqcup \mathtt{s}\ i_1) &= i_2 \sqcup \mathtt{s}\ (\mathtt{s}\ i_1) \\
i_1 \sqcup i_2 \sqcup i_3 &= \mathtt{s}\ i_1
\end{aligned}
$$

The unification algorithm

Let $C = \{i_1 \sqcup \mathbf{s}\ (\mathbf{s}\ i_2 \sqcup \mathbf{s}\ i_1) = i_2 \sqcup \mathbf{s}\ (\mathbf{s}\ i_1), \quad i_1 \sqcup i_2 \sqcup i_3 = \mathbf{s}\ i_1\}$

$$\mathbf{z} \sqcup i_1 \sqcup i_2 = \mathbf{z} \sqcup i_1$$

$$\mathbf{z} \sqcup i_2 \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1$$

The unification algorithm

Let $C = \{i_1 \sqcup \mathtt{s}\ (\mathtt{s}\ i_2 \sqcup \mathtt{s}\ i_1) = i_2 \sqcup \mathtt{s}\ (\mathtt{s}\ i_1),\quad i_1 \sqcup i_2 \sqcup i_3 = \mathtt{s}\ i_1\}$

$$\mathtt{z} \sqcup i_1 \sqcup i_2 = \mathtt{z} \sqcup i_1 \qquad \Longrightarrow\ i_1 \mapsto \mathtt{z} \sqcup i_1' \sqcup i_2 \text{ for } i_1' \text{ fresh}$$

$$\mathtt{z} \sqcup i_2 \sqcup i_3 = \mathtt{s}\ \mathtt{z} \sqcup \mathtt{s}\ i_1$$

The unification algorithm

Let $C = \{i_1 \sqcup \mathbf{s}\ (\mathbf{s}\ i_2 \sqcup \mathbf{s}\ i_1) = i_2 \sqcup \mathbf{s}\ (\mathbf{s}\ i_1), \quad i_1 \sqcup i_2 \sqcup i_3 = \mathbf{s}\ i_1\}$

$$\begin{aligned} &\mathbf{z} \sqcup i_1 \sqcup i_2 = \mathbf{z} \sqcup i_1 && \Longrightarrow\ i_1 \mapsto \mathbf{z} \sqcup i_1' \sqcup i_2 \text{ for } i_1' \text{ fresh} \\ &\mathbf{z} \sqcup i_2 \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1 && \end{aligned}$$

$$\mathbf{z} \sqcup i_2 \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ (\mathbf{z} \sqcup i_1' \sqcup i_2)$$

The unification algorithm

Let $C = \{i_1 \sqcup \mathbf{s}\ (\mathbf{s}\ i_2 \sqcup \mathbf{s}\ i_1) = i_2 \sqcup \mathbf{s}\ (\mathbf{s}\ i_1),\quad i_1 \sqcup i_2 \sqcup i_3 = \mathbf{s}\ i_1\}$

$$\mathbf{z} \sqcup i_1 \sqcup i_2 = \mathbf{z} \sqcup i_1 \qquad \Longrightarrow \quad i_1 \mapsto \mathbf{z} \sqcup i_1' \sqcup i_2 \text{ for } i_1' \text{ fresh}$$

$$\mathbf{z} \sqcup i_2 \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1$$

$$\mathbf{z} \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1' \sqcup \mathbf{s}\ i_2$$

The unification algorithm

Let $C = \{i_1 \sqcup \mathbf{s}\ (\mathbf{s}\ i_2 \sqcup \mathbf{s}\ i_1) = i_2 \sqcup \mathbf{s}\ (\mathbf{s}\ i_1), \quad i_1 \sqcup i_2 \sqcup i_3 = \mathbf{s}\ i_1\}$

$$\mathbf{z} \sqcup i_1 \sqcup i_2 = \mathbf{z} \sqcup i_1 \qquad \Longrightarrow \quad i_1 \mapsto \mathbf{z} \sqcup i_1' \sqcup i_2 \text{ for } i_1' \text{ fresh}$$

$$\mathbf{z} \sqcup i_2 \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1$$

$$\mathbf{z} \sqcup i_3 = \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1' \sqcup \mathbf{s}\ i_2 \qquad \Longrightarrow \quad i_3 \mapsto \mathbf{s}\ \mathbf{z} \sqcup \mathbf{s}\ i_1' \sqcup \mathbf{s}\ i_2$$

The unification algorithm

Let $C = \{i_1 \sqcup \mathtt{s}\ (\mathtt{s}\ i_2 \sqcup \mathtt{s}\ i_1) = i_2 \sqcup \mathtt{s}\ (\mathtt{s}\ i_1),\quad i_1 \sqcup i_2 \sqcup i_3 = \mathtt{s}\ i_1\}$

$$\mathtt{z} \sqcup i_1 \sqcup i_2 = \mathtt{z} \sqcup i_1 \qquad \Longrightarrow \quad i_1 \mapsto \mathtt{z} \sqcup i_1' \sqcup i_2 \text{ for } i_1' \text{ fresh}$$

$$\mathtt{z} \sqcup i_2 \sqcup i_3 = \mathtt{s}\ \mathtt{z} \sqcup \mathtt{s}\ i_1$$

$$\mathtt{z} \sqcup i_3 = \mathtt{s}\ \mathtt{z} \sqcup \mathtt{s}\ i_1' \sqcup \mathtt{s}\ i_2 \qquad \Longrightarrow \quad i_3 \mapsto \mathtt{s}\ \mathtt{z} \sqcup \mathtt{s}\ i_1' \sqcup \mathtt{s}\ i_2$$

Most general unifier $\theta = i_1 \mapsto \mathtt{z} \sqcup i_1' \sqcup i_2,\quad i_3 \mapsto \mathtt{s}\ \mathtt{z} \sqcup \mathtt{s}\ i_1' \sqcup \mathtt{s}\ i_2$

The unification algorithm

(Trivial)	$\{l = l\} \cup C; \theta \rightsquigarrow C; \theta$	
(Orient)	$\{l = l'\} \cup C; \theta \rightsquigarrow \{l' = l\} \cup C; \theta$	if $l' = \mathbf{z}$ or $\mathbf{z} \sqcup i$
(Eliminate 1)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow \widehat{C\{l/i\}}; \widehat{\theta\{l/i\}}, i \mapsto l$	if $i \notin l$
(Eliminate 2)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow$	if $\mathbf{s}^m\ i \in l$ with $m = 0$
	let $l' = l\{i'/i\}$ in $\widehat{C\{l'/i\}}; \widehat{\theta\{l'/i\}}, i \mapsto l'$	for some fresh i'
(Decompose)	$\{\mathbf{z} = \mathbf{z} \sqcup (\sqcup_{i \in V}\ i)\} \cup C; \theta \rightsquigarrow \{\mathbf{z} \sqcup i = \mathbf{z}\}_{i \in V} \cup C; \theta$	
(Clash)	$\{\mathbf{z} = l\} \cup C; \theta \rightsquigarrow \bot$	if $\mathbf{s} \in l$

Three outcomes We get an mgu, there is not solution, or we're stuck

The unification algorithm

(**Trivial**)	$\{l = l\} \cup C; \theta \rightsquigarrow C; \theta$	
(**Orient**)	$\{l = l'\} \cup C; \theta \rightsquigarrow \{l' = l\} \cup C; \theta$	if $l' = \mathbf{z}$ or $\mathbf{z} \sqcup i$
(**Eliminate 1**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow \widehat{C\{l/i\}}; \widehat{\theta\{l/i\}}, i \mapsto l$	if $i \notin l$
(**Eliminate 2**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow$	if $\mathbf{s}^m\ i \in l$ with $m = 0$
	let $l' = l\{i'/i\}$ in $\widehat{C\{l'/i\}}; \widehat{\theta\{l'/i\}}, i \mapsto l'$	for some fresh i'
(**Decompose**)	$\{\mathbf{z} = \mathbf{z} \sqcup (\sqcup_{i \in V}\ i)\} \cup C; \theta \rightsquigarrow \{\mathbf{z} \sqcup i = \mathbf{z}\}_{i \in V} \cup C; \theta$	
(**Clash**)	$\{\mathbf{z} = l\} \cup C; \theta \rightsquigarrow \bot$	if $\mathbf{s} \in l$

Three outcomes We get an mgu, there is not solution, or we're stuck

Theorem If $C; - \rightsquigarrow^* \emptyset; \theta$, then θ is a m.g.u.

The unification algorithm

(**Trivial**)	$\{l = l\} \cup C; \theta \rightsquigarrow C; \theta$	
(**Orient**)	$\{l = l'\} \cup C; \theta \rightsquigarrow \{l' = l\} \cup C; \theta$	if $l' = \mathbf{z}$ or $\mathbf{z} \sqcup i$
(**Eliminate 1**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow \widehat{C\{l/i\}}; \widehat{\theta\{l/i\}}, i \mapsto l$	if $i \notin l$
(**Eliminate 2**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow$	if $\mathbf{s}^m\ i \in l$ with $m = 0$
	let $l' = l\{i'/i\}$ in $\widehat{C\{l'/i\}}; \widehat{\theta\{l'/i\}}, i \mapsto l'$	for some fresh i'
(**Decompose**)	$\{\mathbf{z} = \mathbf{z} \sqcup (\sqcup_{i \in V}\ i)\} \cup C; \theta \rightsquigarrow \{\mathbf{z} \sqcup i = \mathbf{z}\}_{i \in V} \cup C; \theta$	
(**Clash**)	$\{\mathbf{z} = l\} \cup C; \theta \rightsquigarrow \bot$	if $\mathbf{s} \in l$

Three outcomes We get an mgu, there is not solution, or we're stuck

Theorem If $C; - \rightsquigarrow^* \emptyset; \theta$, then θ is a m.g.u. (but the converse does not hold)

The unification algorithm

(**Trivial**)	$\{l = l\} \cup C; \theta \rightsquigarrow C; \theta$	
(**Orient**)	$\{l = l'\} \cup C; \theta \rightsquigarrow \{l' = l\} \cup C; \theta$	if $l' = \mathbf{z}$ or $\mathbf{z} \sqcup i$
(**Eliminate 1**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow \widehat{C\{l/i\}}; \widehat{\theta\{l/i\}}, i \mapsto l$	if $i \notin l$
(**Eliminate 2**)	$\{\mathbf{z} \sqcup i = l\} \cup C; \theta \rightsquigarrow$	if $\mathbf{s}^m\ i \in l$ with $m = 0$
	let $l' = l\{i'/i\}$ in $\widehat{C\{l'/i\}}; \widehat{\theta\{l'/i\}}, i \mapsto l'$	for some fresh i'
(**Decompose**)	$\{\mathbf{z} = \mathbf{z} \sqcup (\sqcup_{i \in V}\ i)\} \cup C; \theta \rightsquigarrow \{\mathbf{z} \sqcup i = \mathbf{z}\}_{i \in V} \cup C; \theta$	
(**Clash**)	$\{\mathbf{z} = l\} \cup C; \theta \rightsquigarrow \bot$	if $\mathbf{s} \in l$

Three outcomes We get an mgu, there is not solution, or we're stuck

Theorem If $C; - \rightsquigarrow^* \emptyset; \theta$, then θ is a m.g.u. (but the converse does not hold)

If algorithm gets stuck, heuristics try to find *some* unifier

Predicativize & translating Matita's library to Agda

Predicativize

Translation implemented on the tool **Predicativize**, built over **DkCheck**

System-independent Does not depend on codebase of any other proof assistant

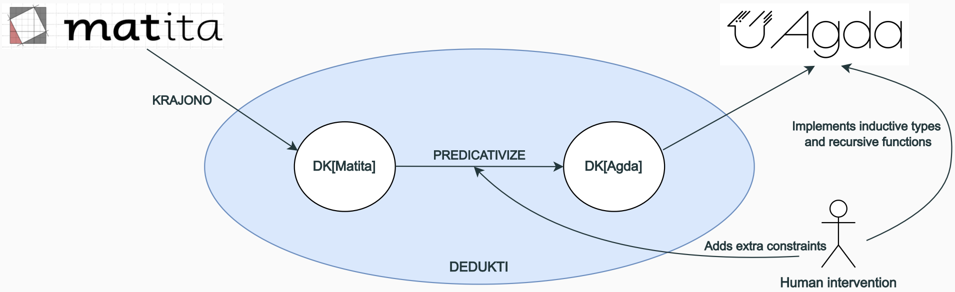
Predicativize

Translation implemented on the tool **Predicativize**, built over **DkCheck**

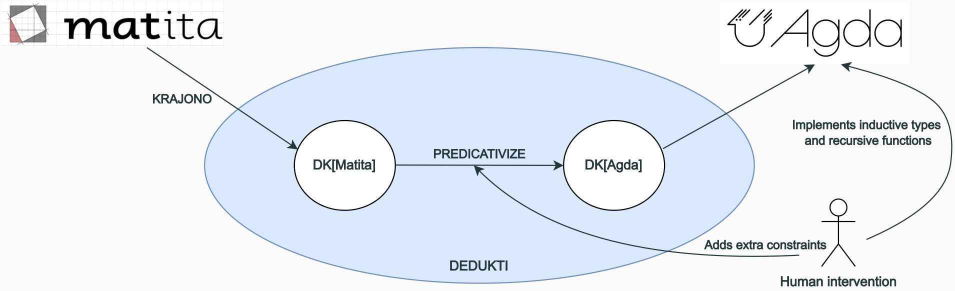
System-independent Does not depend on codebase of any other proof assistant

- Used added constraints
- Translation of rewrite rules
- Agda output

Translating Matita's arithmetic library in Agda



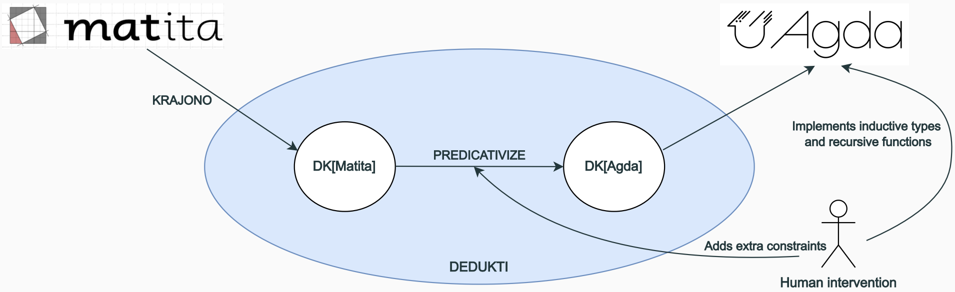
Translating Matita's arithmetic library in Agda



Proofs of Fermat's Little Theorem, Bertrand's Postulate, Pigeonhole Principle, Binomial Law, the Chinese Remainder Theorem, etc in (safe) Agda

Available at `https://github.com/thiagofelicissimo/matita_lib_in_agda`

Translating Matita's arithmetic library in Agda



Proofs of Fermat's Little Theorem, Bertrand's Postulate, Pigeonhole Principle, Binomial Law, the Chinese Remainder Theorem, etc in (safe) Agda

Available at `https://github.com/thiagofelicissimo/matita_lib_in_agda`

Thank you!