Deducteam Type Universe Seminar

Yoan Géran, Rishikesh Vaishnav, Thiago Felicissimo

September 29, 2023

Universes

0 has type Nat, but what is the type of Nat?

Universes

0 has type Nat, but what is the type of Nat?
Some universe U

Universes

0 has type Nat, but what is the type of Nat?
Some universe U

In type theory, (small) types can be given the type of a universe

Universes

0 has type Nat, but what is the type of Nat?
Some universe U

In type theory, (small) types can be given the type of a universe

Many flavours: predicative/impredicative, cumulative/non-cumulative, etc

Universes

0 has type Nat, but what is the type of Nat?
Some universe U

In type theory, (small) types can be given the type of a universe

Many flavours: predicative/impredicative, cumulative/non-cumulative, etc

In many proof assistants: Coq, Agda, Lean, Matita

Universes

0 has type Nat, but what is the type of Nat?
Some universe U

In type theory, (small) types can be given the type of a universe

Many flavours: predicative/impredicative, cumulative/non-cumulative, etc

In many proof assistants: Coq, Agda, Lean, Matita

This talk How to define them in Dedukti

Universe styles in a logical framework

$\mathsf{Ty} : \mathtt{TYPE}$ $\qquad (\llbracket A \text{ type} \rrbracket := \llbracket A \rrbracket : \mathsf{Ty})$

$\mathsf{Tm} : \mathsf{Ty} \to \mathtt{TYPE}$ $\qquad (\llbracket t : A \rrbracket := \llbracket t \rrbracket : \mathsf{Tm}\ \llbracket A \rrbracket)$

Universe styles in a logical framework

$$\mathsf{Ty} : \mathtt{TYPE} \qquad (\llbracket A \text{ type} \rrbracket := \llbracket A \rrbracket : \mathsf{Ty})$$

$$\mathsf{Tm} : \mathsf{Ty} \to \mathtt{TYPE} \qquad (\llbracket t : A \rrbracket := \llbracket t \rrbracket : \mathsf{Tm}\ \llbracket A \rrbracket)$$

Tarski style

$\mathsf{U} : \mathsf{Ty}$

$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \to \mathsf{Ty}$

$\mathsf{u} : \mathsf{Tm}\ \mathsf{U}$

$\mathsf{El}\ \mathsf{u} \longrightarrow \mathsf{U}$

Universe styles in a logical framework

$$\mathsf{Ty} : \mathtt{TYPE} \qquad\qquad (\llbracket A \text{ type} \rrbracket := \llbracket A \rrbracket : \mathsf{Ty})$$

$$\mathsf{Tm} : \mathsf{Ty} \to \mathtt{TYPE} \qquad\qquad (\llbracket t : A \rrbracket := \llbracket t \rrbracket : \mathsf{Tm}\ \llbracket A \rrbracket)$$

Tarski style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \to \mathsf{Ty}$$
$$\mathsf{u} : \mathsf{Tm}\ \mathsf{U}$$
$$\mathsf{El}\ \mathsf{u} \longrightarrow \mathsf{U}$$

Coquand style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \to \mathsf{Ty}$$
$$\mathsf{c} : \mathsf{Ty} \to \mathsf{Tm}\ \mathsf{U}$$
$$\mathsf{El}\ (\mathsf{c}\ A) \longrightarrow A$$
$$\mathsf{c}\ (\mathsf{El}\ A) \longrightarrow A$$

Universe styles in a logical framework

$$\mathsf{Ty} : \mathtt{TYPE} \qquad (\llbracket A \text{ type} \rrbracket := \llbracket A \rrbracket : \mathsf{Ty})$$

$$\mathsf{Tm} : \mathsf{Ty} \to \mathtt{TYPE} \qquad (\llbracket t : A \rrbracket := \llbracket t \rrbracket : \mathsf{Tm}\ \llbracket A \rrbracket)$$

Tarski style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \to \mathsf{Ty}$$
$$\mathsf{u} : \mathsf{Tm}\ \mathsf{U}$$
$$\mathsf{El}\ \mathsf{u} \longrightarrow \mathsf{U}$$

Coquand style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \simeq \mathsf{Ty} : \mathsf{c}$$

Universe styles in a logical framework

$$\mathsf{Ty} : \mathtt{TYPE} \qquad (\llbracket A \text{ type} \rrbracket := \llbracket A \rrbracket : \mathsf{Ty})$$

$$\mathsf{Tm} : \mathsf{Ty} \to \mathtt{TYPE} \qquad (\llbracket t : A \rrbracket := \llbracket t \rrbracket : \mathsf{Tm}\ \llbracket A \rrbracket)$$

Tarski style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \to \mathsf{Ty}$$
$$\mathsf{u} : \mathsf{Tm}\ \mathsf{U}$$
$$\mathsf{El}\ \mathsf{u} \longrightarrow \mathsf{U}$$

Coquand style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \simeq \mathsf{Ty} : \mathsf{c}$$

Russell style

$$\mathsf{U} : \mathsf{Ty}$$
$$\mathsf{Tm}\ \mathsf{U} \longrightarrow \mathsf{Ty}$$

Universe styles in a logical framework

$$\mathsf{Ty} : \mathtt{TYPE} \qquad (\llbracket A\ \mathsf{type} \rrbracket := \llbracket A \rrbracket : \mathsf{Ty})$$

$$\mathsf{Tm} : \mathsf{Ty} \to \mathtt{TYPE} \qquad (\llbracket t : A \rrbracket := \llbracket t \rrbracket : \mathsf{Tm}\ \llbracket A \rrbracket)$$

Tarski style

$\mathsf{U} : \mathsf{Ty}$

$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \to \mathsf{Ty}$

$\mathsf{u} : \mathsf{Tm}\ \mathsf{U}$

$\mathsf{El}\ \mathsf{u} \longrightarrow \mathsf{U}$

Coquand style

$\mathsf{U} : \mathsf{Ty}$

$\mathsf{El} : \mathsf{Tm}\ \mathsf{U} \simeq \mathsf{Ty} : \mathsf{c}$

Russell style

$\mathsf{U} : \mathsf{Ty}$

$\mathsf{Tm}\ \mathsf{U} \longrightarrow \mathsf{Ty}$

In the Dedukti literature, we often use Russell style and change names

Ty	$\rightsquigarrow$	U
Tm	$\rightsquigarrow$	El
U	$\rightsquigarrow$	u

Universe hierarchies

$u : U$ causes inconsistencies

Solution Stratify universes into an hierarchy

Universe hierarchies

$\mathsf{u} : \mathsf{U}$ causes inconsistencies

Solution Stratify universes into an hierarchy

$$\mathsf{U}_s : \mathtt{TYPE}$$
$$\mathsf{El}_s : \mathsf{U}_s \to \mathtt{TYPE} \qquad \text{for } s \in \mathcal{S}$$

Universe hierarchies

$\mathsf{u} : \mathsf{U}$ causes inconsistencies

Solution Stratify universes into an hierarchy

$$\mathsf{U}_s : \mathtt{TYPE}$$
$$\mathsf{El}_s : \mathsf{U}_s \to \mathtt{TYPE} \qquad \text{for } s \in \mathcal{S}$$

$$\mathsf{u}_s : \mathsf{U}_{s'}$$
$$\mathsf{El}_{s'}\ \mathsf{u}_s \longrightarrow \mathsf{U}_s \qquad \text{for } (s, s') \in \mathcal{A}$$

Universe hierarchies

$\mathsf{u} : \mathsf{U}$ causes inconsistencies

Solution Stratify universes into an hierarchy

$$\mathsf{U}_s : \mathtt{TYPE}$$
$$\mathsf{El}_s : \mathsf{U}_s \to \mathtt{TYPE} \qquad \text{for } s \in \mathcal{S}$$

$$\mathsf{u}_s : \mathsf{U}_{s'}$$
$$\mathsf{El}_{s'}\ \mathsf{u}_s \longrightarrow \mathsf{U}_s \qquad \text{for } (s, s') \in \mathcal{A}$$

$$\pi_{s,s'} : (A : \mathsf{U}_s) \to (B : \mathsf{El}_s\ A \to \mathsf{U}_{s'}) \to \mathsf{U}_{s''}$$
$$\mathsf{El}_{s''}\ (\pi_{s,s'}\ A\ B) \longrightarrow (x : \mathsf{El}_s\ A) \to \mathsf{El}_{s'}\ (B\ x) \qquad \text{for } (s, s', s'') \in \mathcal{R}$$

Universe hierarchies

$\mathsf{u} : \mathsf{U}$ causes inconsistencies

Solution Stratify universes into an hierarchy

$$\mathsf{U}_s : \mathtt{TYPE}$$
$$\mathsf{El}_s : \mathsf{U}_s \to \mathtt{TYPE} \qquad \text{for } s \in \mathcal{S}$$

$$\mathsf{u}_s : \mathsf{U}_{s'}$$
$$\mathsf{El}_{s'}\ \mathsf{u}_s \longrightarrow \mathsf{U}_s \qquad \text{for } (s, s') \in \mathcal{A}$$

$$\pi_{s,s'} : (A : \mathsf{U}_s) \to (B : \mathsf{El}_s\ A \to \mathsf{U}_{s'}) \to \mathsf{U}_{s''}$$
$$\mathsf{El}_{s''}\ (\pi_{s,s'}\ A\ B) \longrightarrow (x : \mathsf{El}_s\ A) \to \mathsf{El}_{s'}\ (B\ x) \qquad \text{for } (s, s', s'') \in \mathcal{R}$$

Finite encoding?

Universe hierarchies, finitely

$\mathcal{S} : \texttt{TYPE}$

$\mathcal{A} : \mathcal{S} \to \mathcal{S}$

$\mathcal{R} : \mathcal{S} \to \mathcal{S} \to \mathcal{S}$

...

Universe hierarchies, finitely

$\mathcal{S} : \texttt{TYPE}$

$\mathcal{A} : \mathcal{S} \to \mathcal{S}$

$\mathcal{R} : \mathcal{S} \to \mathcal{S} \to \mathcal{S}$

...

$\mathsf{U} : \mathcal{S} \to \texttt{TYPE}$

$\mathsf{El} : (s : \mathcal{S}) \to \mathsf{U}\ s \to \texttt{TYPE}$

$\mathsf{u} : (s : \mathcal{S}) \to \mathsf{U}\ (\mathcal{A}\ s)$

$\mathsf{El}\ \_\ (\mathsf{u}\ s) \longrightarrow \mathsf{U}\ s$

$\pi : (s\ s' : \mathcal{S}) \to (A : \mathsf{U}\ s) \to (B : \mathsf{El}\ s\ A \to \mathsf{U}\ s') \to \mathsf{U}\ (\mathcal{R}\ s\ s')$

$\mathsf{El}\ \_\ (\pi\ s\ s'\ A\ B) \longrightarrow (x : \mathsf{El}\ s\ A) \to \mathsf{El}\ s'\ (B\ x)$

Introduction: Universe Polymorphism

Universe Polymorphism

Sometimes one wishes to use a definition at multiple universes (e.g. id Nat but also id U).

.

Universe Polymorphism

Sometimes one wishes to use a definition at multiple universes (e.g. id Nat but also id U).

Bad solution. Define a new id_s for each universe U_s.

Universe Polymorphism

Sometimes one wishes to use a definition at multiple universes (e.g. id Nat but also id U).

Bad solution. Define a new id_s for each universe U_s.

Universe polymorphism allows definitions that can be used at multiple universes

$$\mathsf{id}_i : \Pi A : \mathsf{U}_i . A \to A := \lambda A\ x.x$$

Universe Polymorphism

Sometimes one wishes to use a definition at multiple universes (e.g. id Nat but also id U).

Bad solution. Define a new id_s for each universe U_s.

Universe polymorphism allows definitions that can be used at multiple universes

$$\mathsf{id}_i : \Pi A : \mathsf{U}_i . A \to A := \lambda A\ x.x$$

We have id_0 Nat $0 = 0$ and id_1 U_0 Nat $=$ Nat

Universe Polymorphism

Sometimes one wishes to use a definition at multiple universes (e.g. id Nat but also id U).

Bad solution. Define a new id_s for each universe U_s.

Universe polymorphism allows definitions that can be used at multiple universes

$$\mathsf{id}_i : \Pi A : \mathsf{U}_i . A \to A := \lambda A\ x.x$$

We have id_0 Nat $0 = 0$ and id_1 U_0 Nat $=$ Nat

In Dedukti Level ($=$ sort) quantification can be simulated directly by framework's function type

Universe Polymorphism

Sometimes one wishes to use a definition at multiple universes (e.g. id Nat but also id U).

Bad solution. Define a new id_s for each universe U_s.

Universe polymorphism allows definitions that can be used at multiple universes

$$\mathsf{id}_i : \Pi A : \mathsf{U}_i . A \to A := \lambda A\ x . x$$

We have id_0 Nat $0 = 0$ and id_1 U_0 Nat $=$ Nat

In Dedukti Level ($=$ sort) quantification can be simulated directly by framework's function type

However, often we require levels to satisfy a specific equational theory.
This is the hard part

Predicative Universe Polymorphism

Predicative levels

Predicative levels

$$l, l' ::= i \mid 0 \mid \mathsf{S}\ l \mid l \sqcup l'$$

with equality defined by

$$l \simeq l' \qquad \text{iff} \qquad \forall \sigma : \mathcal{V} \to \mathbb{N}.\ [\![l]\!]_\sigma = [\![l']\!]_\sigma$$

where $[\![-]\!]_\sigma$ interprets levels in obvious way.

Predicative levels

Predicative levels

$$l, l' ::= i \mid 0 \mid \mathsf{S}\ l \mid l \sqcup l'$$

with equality defined by

$$l \simeq l' \qquad \text{iff} \qquad \forall \sigma : \mathcal{V} \to \mathbb{N}.\ [\![l]\!]_\sigma = [\![l']\!]_\sigma$$

where $[\![-]\!]_\sigma$ interprets levels in obvious way.

Problem How to encode $\simeq$ in Dedukti?

Solutions

Genestier 20 Rewrite system to decide $\simeq$
Based on existence of canonical forms for levels
Requires AC matching and AC equivalence

Solutions

Genestier 20 Rewrite system to decide $\simeq$
Based on existence of canonical forms for levels
Requires AC matching and AC equivalence

Blanqui 22 ~~AC matching~~ normalized rewriting

$$x \sqcup y \sqcup x \simeq x \sqcup x \sqcup y \longrightarrow x \sqcup y$$

Solutions

Genestier 20 Rewrite system to decide $\simeq$
Based on existence of canonical forms for levels
Requires AC matching and AC equivalence

Blanqui 22 ~~AC matching~~ normalized rewriting

$$x \sqcup y \sqcup x \simeq x \sqcup x \sqcup y \longrightarrow x \sqcup y$$

Felicissimo 23 Abandon idea of encoding $\simeq$ with rewriting
We have $\simeq \cdot \longrightarrow \subseteq \longrightarrow \cdot \simeq$, so can postpone $\simeq$ to end of conversion check
~~AC matching/normalized rewriting~~ syntactic matching + decide $\simeq$

Solutions

Genestier 20 Rewrite system to decide $\simeq$
Based on existence of canonical forms for levels
Requires AC matching and AC equivalence

Blanqui 22 ~~AC matching~~ normalized rewriting

$$x \sqcup y \sqcup x \simeq x \sqcup x \sqcup y \longrightarrow x \sqcup y$$

Felicissimo 23 Abandon idea of encoding $\simeq$ with rewriting
We have $\simeq \cdot \longrightarrow \subseteq \longrightarrow \cdot \simeq$, so can postpone $\simeq$ to end of conversion check
~~AC matching/normalized rewriting~~ syntactic matching + decide $\simeq$
If Dedukti+AC is ok, why not Dedukti+E for arbitrary E?

Solutions

Genestier 20 Rewrite system to decide $\simeq$
Based on existence of canonical forms for levels
Requires AC matching and AC equivalence

Blanqui 22 ~~AC matching~~ normalized rewriting

$$x \sqcup y \sqcup x \simeq x \sqcup x \sqcup y \longrightarrow x \sqcup y$$

Felicissimo 23 Abandon idea of encoding $\simeq$ with rewriting
We have $\simeq \cdot \longrightarrow \subseteq \longrightarrow \cdot \simeq$, so can postpone $\simeq$ to end of conversion check
~~AC matching/normalized rewriting~~ syntactic matching + decide $\simeq$
If Dedukti+AC is ok, why not Dedukti+E for arbitrary E?

Takeaway message No way to encode in vanilla Dedukti

Solutions

Genestier 20 Rewrite system to decide $\simeq$
Based on existence of canonical forms for levels
Requires AC matching and AC equivalence

Blanqui 22 ~~AC matching~~ normalized rewriting

$$x \sqcup y \sqcup x \simeq x \sqcup x \sqcup y \longrightarrow x \sqcup y$$

Felicissimo 23 Abandon idea of encoding $\simeq$ with rewriting
We have $\simeq \cdot \longrightarrow \subseteq \longrightarrow \cdot \simeq$, so can postpone $\simeq$ to end of conversion check
~~AC matching/normalized rewriting~~ syntactic matching + decide $\simeq$
If Dedukti+AC is ok, why not Dedukti+E for arbitrary E?

Takeaway message No way to encode in vanilla Dedukti
Moreover, to show confluence, all 3 options require confinement or showing SN before confluence (reason: non-left-linear rules)

Impredicative Universe Normalization

Introduction

- In theorem provers like Lean and Coq, we have an infinite universe hierarchy starting with the base universe (`Sort 0`) which is reserved for propositions.

Introduction

- In theorem provers like Lean and Coq, we have an infinite universe hierarchy starting with the base universe (`Sort 0`) which is reserved for propositions.
- We must encode universe impredicativity in the context of polymorphic types deriving from the rule:

$$\frac{\Gamma \vdash A : \mathsf{U}_{\ell} \qquad \Gamma, x : A \vdash B : \mathsf{U}_{\ell'}}{\Gamma \vdash \forall x : A.\ B : \mathsf{U}_{\mathtt{i}(\ell, \ell')}}$$

Introduction

- In theorem provers like Lean and Coq, we have an infinite universe hierarchy starting with the base universe (`Sort 0`) which is reserved for propositions.
- We must encode universe impredicativity in the context of polymorphic types deriving from the rule:

$$\frac{\Gamma \vdash A : \mathsf{U}_{\ell} \qquad \Gamma, x : A \vdash B : \mathsf{U}_{\ell'}}{\Gamma \vdash \forall x : A.\ B : \mathsf{U}_{\mathtt{i}(\ell,\ell')}}$$

where `i` (i.e. `imax`, "impredicative max") has the semantics:

$$\mathtt{i}(\ell,\ell') = \begin{cases} 0, & \text{if } \ell' = 0 \\ \max(\ell,\ell'), & \text{otherwise.} \end{cases}$$

- In theorem provers like Lean and Coq, we have an infinite universe hierarchy starting with the base universe (`Sort 0`) which is reserved for propositions.
- We must encode universe impredicativity in the context of polymorphic types deriving from the rule:

$$\frac{\Gamma \vdash A : \mathsf{U}_{\ell} \qquad \Gamma, x : A \vdash B : \mathsf{U}_{\ell'}}{\Gamma \vdash \forall x : A.\, B : \mathsf{U}_{\mathtt{i}(\ell,\ell')}}$$

where `i` (i.e. `imax`, "impredicative max") has the semantics:

$$\mathtt{i}(\ell,\ell') = \begin{cases} 0, & \text{if } \ell' = 0 \\ \max(\ell,\ell'), & \text{otherwise.} \end{cases}$$

- In total, we have the following grammar for universe terms:

$$\ell := 0 \mid \mathtt{s}\,(\ell) \mid \mathtt{m}(\ell,\ell') \mid \mathtt{i}(\ell,\ell') \mid x$$

where x is from a countable set of variables $\mathcal{X}$.
We denote this set of terms by $\mathcal{L}$.

- For a valuation $\sigma : \mathcal{X} \to \mathbb{N}$ we define the value $[\![\ell]\!]_\sigma$ of a level term ℓ according to the rules:

$$[\![0]\!]_\sigma = 0 \qquad [\![\mathtt{s}\,(t)]\!]_\sigma = \mathtt{s}\,([\![t]\!]_\sigma) \qquad [\![x]\!]_\sigma = \sigma(t)$$

$$[\![\mathtt{m}(\ell, \ell')]\!]_\sigma = \max([\![\ell]\!]_\sigma, [\![\ell']\!]_\sigma)$$

$$[\![\mathtt{i}(\ell, \ell')]\!]_\sigma = \begin{cases} 0, & \text{if } [\![\ell']\!]_\sigma = 0 \\ \max([\![\ell]\!]_\sigma, [\![\ell']\!]_\sigma), & \text{otherwise.} \end{cases}$$

Introduction

- For a valuation $\sigma : \mathcal{X} \to \mathbb{N}$ we define the value $[\![\ell]\!]_\sigma$ of a level term ℓ according to the rules:

$$[\![0]\!]_\sigma = 0 \qquad [\![\mathtt{s}\,(t)]\!]_\sigma = \mathtt{s}\,([\![t]\!]_\sigma) \qquad [\![x]\!]_\sigma = \sigma(t)$$

$$[\![\mathtt{m}(\ell, \ell')]\!]_\sigma = \max([\![\ell]\!]_\sigma, [\![\ell']\!]_\sigma)$$

$$[\![\mathtt{i}(\ell, \ell')]\!]_\sigma = \begin{cases} 0, & \text{if } [\![\ell']\!]_\sigma = 0 \\ \max([\![\ell]\!]_\sigma, [\![\ell']\!]_\sigma), & \text{otherwise.} \end{cases}$$

- We define semantic relations between universe terms:

$$\ell =_{[\![]\!]} \ell' \iff \text{for all } \sigma : \mathcal{X} \to \mathbb{N},\ [\![\ell]\!]_\sigma = [\![\ell']\!]_\sigma$$

$$\ell \leq_{[\![]\!]} \ell' \iff \text{for all } \sigma : \mathcal{X} \to \mathbb{N},\ [\![\ell]\!]_\sigma \leq [\![\ell']\!]_\sigma$$

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

becomes

$$\color{red}{1 +}\ \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no $\mathtt{i}$) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" $\mathtt{s}$'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

with all subterms incomparable.

- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

becomes

$$\mathtt{m}(2 + x, 1 + \mathtt{m}(\mathtt{m}(5, x), y))$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{m}(2 + x, 1 + \mathtt{m}(\mathtt{m}(5, x), y))$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{m}(2 + x, \mathtt{m}(1 + \mathtt{m}(5, x), 1 + y))$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{m}(2 + x, \mathtt{m}(1 + \mathtt{m}(5, x), 1 + y))$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{m}(2 + x, \mathtt{m}(\mathtt{m}({\color{red}6}, {\color{red}1 +} x), 1 + y))$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{m}\left(2 + x, \mathtt{m}\left(\mathtt{m}\left(6, 1 + x\right), 1 + y\right)\right)$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{maxS}\,(2 + x,\ 1 + x,\ 1 + y,\ 6)$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{maxS}(\underbrace{2 + x,\ 1 + x}_{\geq_{[\![\,]\!]}},\ 1 + y,\ 6)$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

A predicative normal form

- We can take some inspiration from the normal form introduced by Genestier[1] for the predicative (no `i`) case. Here, we consider "subterms" of the form $n + x$ or n where $n \in \mathbb{N}$ and $x \in \mathcal{X}$.
- We proceed by "pushing in" `s`'s (i.e. constant additions) and eliminating "dominated" subterms until we arrive at the form:

$$\mathtt{maxS}(n_1 + x_1, \ldots, n_k + x_k,\ n),$$

  with all subterms incomparable.
- For example:

$$1 + \mathtt{m}(1 + x, \mathtt{m}(\mathtt{m}(5, x), y))$$

  becomes

$$\mathtt{maxS}(2 + x,\ 1 + y,\ 6)$$

[1] Guillaume Genestier. Encoding Agda Programs Using Rewriting, https://drops.dagstuhl.de/opus/volltexte/2020/12353

Impredicative normal form: goals

- We also want a normal form like $\texttt{maxS}(u_1, \ldots, u_n)$, where the u_i are "minimal" subterms picked from a grammar and semantic such that:

Impredicative normal form: goals

- We also want a normal form like $\mathtt{maxS}(u_1, \ldots, u_n)$, where the u_i are "minimal" subterms picked from a grammar and semantic such that:
  1. (`existence`) they **completely characterize all universe terms**, that is, for all t there exists $\{u_1, \ldots, u_n\}$ such that $t = \mathtt{maxS}(u_1, \ldots, u_n)$.

Impredicative normal form: goals

- We also want a normal form like $\texttt{maxS}(u_1, \ldots, u_n)$, where the u_i are "minimal" subterms picked from a grammar and semantic such that:
  1. ($\texttt{existence}$) they **completely characterize all universe terms**, that is, for all t there exists $\{u_1, \ldots, u_n\}$ such that $t = \texttt{maxS}(u_1, \ldots, u_n)$.
  2. ($\texttt{uniqueness}$) they **uniquely identify a normal form**, such that:

     $$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_m\}$$

     when $\{u_1,\ \ldots,\ u_n\}$ and $\{v_1,\ \ldots,\ v_m\}$ are incomparable.

Impredicative normal form: goals

- We also want a normal form like $\texttt{maxS}(u_1, \ldots, u_n)$, where the u_i are "minimal" subterms picked from a grammar and semantic such that:
  1. ($\texttt{existence}$) they **completely characterize all universe terms**, that is, for all t there exists $\{u_1, \ldots, u_n\}$ such that $t = \texttt{maxS}(u_1, \ldots, u_n)$.
  2. ($\texttt{uniqueness}$) they **uniquely identify a normal form**, such that:

     $$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_m\}$$

     when $\{u_1,\ \ldots,\ u_n\}$ and $\{v_1,\ \ldots,\ v_m\}$ are incomparable.
  3. ($\texttt{attainability}$) they are **easily comparable** via rewrite rules, so we can reduce $\texttt{maxS}(u,\ v)$ into $\texttt{maxS}(u)$ when $v \leq_{[\![]\!]} u$, implying that **a normal form can be practically attained**.

Pulling out `m`/Pushing in `s`

- We plan to produce a normal form consisting of the maximum of a set of subterms. To this end, we must "pull out" the `m` operators until they are no longer nested within any other operator.

Pulling out $\mathtt{m}$/Pushing in $\mathtt{s}$

- We plan to produce a normal form consisting of the maximum of a set of subterms. To this end, we must "pull out" the $\mathtt{m}$ operators until they are no longer nested within any other operator.
- We immediately have $\mathtt{s}\left(\mathtt{m}(x, y)\right) = \mathtt{m}(\mathtt{s}\left(x\right), \mathtt{s}\left(y\right))$. For the $\mathtt{i}$ case we derive the equalities:

$$\mathtt{i}(\mathtt{m}(x, y), z) = \mathtt{m}(\mathtt{i}(x, z), \mathtt{i}(y, z))$$

$\mathcal{L} \to \mathcal{S}_{\mathsf{nf}}^{--}$

Pulling out $\mathtt{m}$/Pushing in $\mathtt{s}$

- We plan to produce a normal form consisting of the maximum of a set of subterms. To this end, we must "pull out" the $\mathtt{m}$ operators until they are no longer nested within any other operator.
- We immediately have $\mathtt{s}\left(\mathtt{m}(x, y)\right) = \mathtt{m}(\mathtt{s}\left(x\right), \mathtt{s}\left(y\right))$. For the $\mathtt{i}$ case we derive the equalities:

$$\mathtt{i}(\mathtt{m}(x, y), z) = \mathtt{m}(\mathtt{i}(x, z), \mathtt{i}(y, z))$$

$$\mathtt{i}(x, \mathtt{m}(y, z)) = \mathtt{m}(\mathtt{i}(x, y), \mathtt{i}(x, z))$$

$\mathcal{L} \to \mathcal{S}_{\mathsf{nf}}^{--}$

Pulling out $\mathtt{m}$/Pushing in $\mathtt{s}$

- We plan to produce a normal form consisting of the maximum of a set of subterms. To this end, we must "pull out" the $\mathtt{m}$ operators until they are no longer nested within any other operator.
- We immediately have $\mathtt{s}\,(\mathtt{m}(x, y)) = \mathtt{m}(\mathtt{s}\,(x)\,, \mathtt{s}\,(y))$. For the $\mathtt{i}$ case we derive the equalities:

$$\mathtt{i}(\mathtt{m}(x, y), z) = \mathtt{m}(\mathtt{i}(x, z), \mathtt{i}(y, z))$$

$$\mathtt{i}(x, \mathtt{m}(y, z)) = \mathtt{m}(\mathtt{i}(x, y), \mathtt{i}(x, z))$$

$\mathcal{L} \to \mathcal{S}_{\mathsf{nf}}^{--}$

- To restrict $\mathtt{s}$ to variables, we can push it into the $\mathtt{i}$ terms according to the equality:

$$\mathtt{s}\,(\mathtt{i}(x, y)) = \mathtt{m}(\mathtt{s}\,(y)\,, \mathtt{i}(\mathtt{s}\,(x)\,, y))$$

$\mathcal{L} \to \mathcal{S}_{\mathsf{nf}}^{--}$

Simplifying `i` subterms: RHS

- We wish to simplify the righthand-side of the `i` operators in our normal form. We can do so by observing the equalities:

Simplifying `i` subterms: RHS

- We wish to simplify the righthand-side of the `i` operators in our normal form. We can do so by observing the equalities:

$$\mathtt{i}(u, \mathtt{i}(v, w)) = \mathtt{m}(\mathtt{i}(u, w), \mathtt{i}(v, w))$$

$\mathcal{L} \to \mathcal{S}^{--}_{\mathsf{nf}}$

which serve to restrict the RHS to variables.

Simplifying $\mathtt{i}$ subterms: RHS

- We wish to simplify the righthand-side of the $\mathtt{i}$ operators in our normal form. We can do so by observing the equalities:

$$\mathcal{L} \to \mathcal{S}^{--}_{\mathsf{nf}}$$

$$\begin{aligned} \mathtt{i}(u, \mathtt{i}(v, w)) &= \mathtt{m}(\mathtt{i}(u, w), \mathtt{i}(v, w)) \\ \mathtt{i}(u, \mathtt{s}\,(v)) &= \mathtt{m}(u, \mathtt{s}\,(v)) \end{aligned}$$

which serve to restrict the RHS to variables.

Simplifying $\mathtt{i}$ subterms: RHS

- We wish to simplify the righthand-side of the $\mathtt{i}$ operators in our normal form. We can do so by observing the equalities:

$\mathcal{L} \to \mathcal{S}_{\mathsf{nf}}^{--}$

$$\begin{aligned} \mathtt{i}(u, \mathtt{i}(v, w)) &= \mathtt{m}(\mathtt{i}(u, w), \mathtt{i}(v, w)) \\ \mathtt{i}(u, \mathtt{s}\,(v)) &= \mathtt{m}(u, \mathtt{s}\,(v)) \\ \mathtt{i}(u, 0) &= 0 \end{aligned}$$

  which serve to restrict the RHS to variables.

Simplifying $\mathtt{i}$ subterms: RHS

- We wish to simplify the righthand-side of the $\mathtt{i}$ operators in our normal form. We can do so by observing the equalities:

$$\boxed{\mathcal{L} \to \mathcal{S}_{\mathsf{nf}}^{--}}$$

$$\begin{aligned} \mathtt{i}(u, \mathtt{i}(v, w)) &= \mathtt{m}(\mathtt{i}(u, w), \mathtt{i}(v, w)) \\ \mathtt{i}(u, \mathtt{s}\,(v)) &= \mathtt{m}(u, \mathtt{s}\,(v)) \\ \mathtt{i}(u, 0) &= 0 \end{aligned}$$

  which serve to restrict the RHS to variables.
- As there are no rules to further simplify the lefthand-side of $\mathtt{i}$, we accept the $\mathtt{s}$, $\mathtt{i}$, and 0 in the LHS of $\mathtt{i}$ as part of our subterms.

A pseudo-pseudo-normal form

- This leads us to a normal form that looks like $\mathtt{maxS}(u_1, \ldots, u_n)$, where the subterms u_i are constructed from the grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}^{--}$.

A pseudo-pseudo-normal form

- This leads us to a normal form that looks like $\mathtt{maxS}(u_1, \ldots, u_n)$, where the subterms u_i are constructed from the grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

  We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}^{--}$.
- However, this normal form is not enough! It does not guarantee `uniqueness` of the representation.

A pseudo-pseudo-normal form

- This leads us to a normal form that looks like $\texttt{maxS}(u_1, \ldots, u_n)$, where the subterms u_i are constructed from the grammar:

$$u := \texttt{s}^n(0) \mid \texttt{s}^n(x) \mid \texttt{i}(u, x).$$

  We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}^{--}$.
- However, this normal form is not enough! It does not guarantee `uniqueness` of the representation.
- For example, we have the equality:

$$\texttt{maxS}(\texttt{i}(x, y), \texttt{i}(y, x)) =_{[\![]\!]} \texttt{maxS}(x, y).$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
    - y is **always** considered as part of the maximum, and so should be its own subterm.

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}(\mathtt{i}(x, y), \mathtt{i}(y, x))$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}(\mathtt{i}(x, y), \mathtt{i}(y, x))$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when** $\mathbf{y \neq 0}$, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}(\mathtt{A}(\{y\}, x), \mathtt{A}(\{\}, y), \mathtt{A}(\{x\}, y), \mathtt{A}(\{\}, x))$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when** $\mathbf{y \neq 0}$, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}(\mathtt{A}(\{y\}, x), \mathtt{A}(\{\}, y), \mathtt{A}(\{x\}, y), \mathtt{A}(\{\}, x))$$

$$\leq_{[\![\,]\!]}$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when** $\mathbf{y \neq 0}$, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}({\color{blue}\mathtt{A}(\{\}, x)}, \mathtt{A}(\{\}, y), \mathtt{A}(\{x\}, y))$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}(\mathtt{A}(\{\}, x), \underbrace{\mathtt{A}(\{\}, y), \mathtt{A}}_{\geq_{[\![\,]\!]}}(\{x\}, y))$$

Deconstructing $\mathtt{i}(x, y)$

- This issue suggests that $\mathtt{i}(x, y)$ can be considered the maximum of simpler component subterms. We can observe that:
  - y is **always** considered as part of the maximum, and so should be its own subterm.
  - x is **only considered when $\mathbf{y \neq 0}$**, so this conditioning should be reflected in its subterm.
- So, we can think of a new subterm of the form $\mathtt{A}(\{y\}, x)$ with the semantic:

$$[\![\mathtt{A}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x), & \text{otherwise.} \end{cases}$$

- With this idea, the previous counterexample is resolved:

$$\mathtt{maxS}(\mathtt{A}(\{\}, x), \mathtt{A}(\{\}, y))$$

Establishing normal form subterms

- However, recall our grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

Establishing normal form subterms

- However, recall our grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

  Note that $\mathtt{i}$'s can be nested, and within an innermost $\mathtt{i}$, the LHS can be of the form $\mathtt{s}^n(0)$ or $\mathtt{s}^n(x)$.

Establishing normal form subterms

- However, recall our grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

  Note that $\mathtt{i}$'s can be nested, and within an innermost $\mathtt{i}$, the LHS can be of the form $\mathtt{s}^n(0)$ or $\mathtt{s}^n(x)$.
- So, we generalize our subterms to the forms $\mathtt{A}(S, x, n)$ and $\mathtt{B}(S, n)$ where

Establishing normal form subterms

- However, recall our grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

  Note that `i`'s can be nested, and within an innermost `i`, the LHS can be of the form $\mathtt{s}^n(0)$ or $\mathtt{s}^n(x)$.

- So, we generalize our subterms to the forms $\mathtt{A}(S, x, n)$ and $\mathtt{B}(S, n)$ where

$$[\![\mathtt{A}(S, x, n)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x) + n, & \text{otherwise.} \end{cases}$$

Establishing normal form subterms

- However, recall our grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

  Note that $\mathtt{i}$'s can be nested, and within an innermost $\mathtt{i}$, the LHS can be of the form $\mathtt{s}^n(0)$ or $\mathtt{s}^n(x)$.

- So, we generalize our subterms to the forms $\mathtt{A}(S, x, n)$ and $\mathtt{B}(S, n)$ where

$$[\![\mathtt{A}(S, x, n)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x) + n, & \text{otherwise.} \end{cases}$$

$$[\![\mathtt{B}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists x \in S, \sigma(x) = 0 \\ n, & \text{otherwise.} \end{cases}$$

.

Establishing normal form subterms

- However, recall our grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{i}(u, x).$$

  Note that $\mathtt{i}$'s can be nested, and within an innermost $\mathtt{i}$, the LHS can be of the form $\mathtt{s}^n(0)$ or $\mathtt{s}^n(x)$.

- So, we generalize our subterms to the forms $\mathtt{A}(S, x, n)$ and $\mathtt{B}(S, n)$ where

$$[\![\mathtt{A}(S, x, n)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists y \in S, \sigma(y) = 0 \\ \sigma(x) + n, & \text{otherwise.} \end{cases}$$

$$[\![\mathtt{B}(S, x)]\!]_\sigma = \begin{cases} 0, & \text{if } \exists x \in S, \sigma(x) = 0 \\ n, & \text{otherwise.} \end{cases}$$

  We also refer to these new subterms as "sublevels".

Establishing normal form subterms

- We can equate terms of the form:

$$\mathtt{i}(\mathtt{i}(\cdots \mathtt{i}(\mathtt{i}(s^n(y), x_1), x_2) \cdots, x_{n-1}), x_n)$$

with:

$$\mathtt{maxS}(\qquad).$$

Establishing normal form subterms

- We can equate terms of the form:

$$\mathtt{i}(\mathtt{i}(\cdots \mathtt{i}(\mathtt{i}(s^n(y), x_1), x_2) \cdots, x_{n-1}), x_n)$$

unguarded

with:

$\mathtt{maxS}($ $\mathtt{A}(\{\}, x_n, 0))$.

Establishing normal form subterms

- We can equate terms of the form:

$$\mathtt{i}(\mathtt{i}(\cdots \mathtt{i}(\mathtt{i}(s^n(y), x_1), x_2) \cdots, x_{n-1}), x_n)$$

with:

guarded by x_n only

$\mathtt{maxS}($ $\mathtt{A}(\{x_n\}, x_{n-1}, 0), \mathtt{A}(\{\}, x_n, 0)).$

Establishing normal form subterms

- We can equate terms of the form:

$$\mathtt{i}(\mathtt{i}(\cdots\mathtt{i}(\mathtt{i}(s^n(y),x_1),x_2)\cdots,x_{n-1}),x_n)$$

with:

guarded by all of $x_2,\ldots,x_n$

$\mathtt{maxS}($ $\mathtt{A}(\{x_2,\ldots,x_n\},x_1,0),\ldots,\mathtt{A}(\{x_n\},x_{n-1},0),\mathtt{A}(\{\},x_n,0)).$

Establishing normal form subterms

- We can equate terms of the form:

$$\mathtt{i}(\mathtt{i}(\cdots \mathtt{i}(\mathtt{i}(s^n(y), x_1), x_2) \cdots, x_{n-1}), x_n)$$

with:

guarded by all x_i

$$\mathtt{maxS}(\mathtt{A}(\{x_1,\ldots,x_n\},y,n),\mathtt{A}(\{x_2,\ldots,x_n\},x_1,0),\ldots,\mathtt{A}(\{x_n\},x_{n-1},0),\mathtt{A}(\{\},x_n,0)).$$

Establishing normal form subterms

- We can equate terms of the form:

$$\mathtt{i}(\mathtt{i}(\cdots \mathtt{i}(\mathtt{i}(s^n(y), x_1), x_2) \cdots, x_{n-1}), x_n) \qquad \boxed{\mathcal{S}_{\mathsf{nf}}^{--} \to \mathcal{S}_{\mathsf{nf}}^{-}}$$

with:

$$\mathtt{maxS}(\mathtt{A}(\{x_1,\ldots,x_n\},y,n),\mathtt{A}(\{x_2,\ldots,x_n\},x_1,0),\ldots,\mathtt{A}(\{x_n\},x_{n-1},0),\mathtt{A}(\{\},x_n,0)).$$

- Similarly,

$$\mathtt{i}(\mathtt{i}(\cdots \mathtt{i}(\mathtt{i}(s^n(0), x_1), x_2) \cdots, x_{n-1}), x_n) \qquad \boxed{\mathcal{S}_{\mathsf{nf}}^{--} \to \mathcal{S}_{\mathsf{nf}}^{-}}$$

becomes:

$$\mathtt{maxS}(\mathtt{B}(\{x_1,\ldots,x_n\},n),\mathtt{A}(\{x_2,\ldots,x_n\},x_1,0),\ldots,\mathtt{A}(\{x_n\},x_{n-1},0),\mathtt{A}(\{\},x_n,0))$$

A pseudo-normal form

- We now have the following subterm grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{A}(\{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n)$$

We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}^{-}$.

A pseudo-normal form

- We now have the following subterm grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{A}(\{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n)$$

We denote this set of subterms by $\mathcal{S}^-_{\mathsf{nf}}$.

- However, this normal form still not sufficient to satisfy the `uniqueness` property. We have the equalities:

$$\mathtt{maxS}(\mathtt{s}^n(0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{B}(\{\}, n))$$

$$\mathtt{maxS}(\mathtt{s}^n(x)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{\}, x, n)),$$

$\mathcal{S}^-_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}$

A pseudo-normal form

- We now have the following subterm grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{A}(\{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n)$$

  We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}^{-}$.
- However, this normal form still not sufficient to satisfy the `uniqueness` property. We have the equalities:

$$\mathtt{maxS}(\mathtt{s}^n(0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{B}(\{\}, n))$$

$\mathcal{S}_{\mathsf{nf}}^{-} \to \mathcal{S}_{\mathsf{nf}}$

$$\mathtt{maxS}(\mathtt{s}^n(x)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{\}, x, n)),$$

  and we also have

$$\mathtt{maxS}(\mathtt{B}(S, 0)) =_{[\![]\!]} \mathtt{maxS}()$$

$\mathcal{S}_{\mathsf{nf}}^{-} \to \mathcal{S}_{\mathsf{nf}}$

  for all sets S (where we interpret $\mathtt{maxS}()$ as 0).

A pseudo-normal form

- We now have the following subterm grammar:

$$u := \mathtt{s}^n(0) \mid \mathtt{s}^n(x) \mid \mathtt{A}(\{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n)$$

  We denote this set of subterms by $\mathcal{S}^-_{\mathsf{nf}}$.
- However, this normal form still not sufficient to satisfy the `uniqueness` property. We have the equalities:

$$\mathtt{maxS}(\mathtt{s}^n(0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{B}(\{\}, n)) \qquad \boxed{\mathcal{S}^-_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}}$$

$$\mathtt{maxS}(\mathtt{s}^n(x)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{\}, x, n)),$$

  and we also have

$$\mathtt{maxS}(\mathtt{B}(S, 0)) =_{[\![]\!]} \mathtt{maxS}() \qquad \boxed{\mathcal{S}^-_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}}$$

  for all sets S (where we interpret $\mathtt{maxS}()$ as 0).
- These equalities, when applied, allow us to restrict to a subterm grammar consisting of sublevels alone:

$$u := \mathtt{A}(\{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n + 1).$$

The true normal form

- However, we still have the following equality:

$$\mathtt{maxS}(\mathtt{A}(\{\}, x, 0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{x\}, x, 0))$$

The true normal form

- However, we still have the following equality:

$$\texttt{maxS}(\texttt{A}(\{\}, x, 0)) =_{[\![]\!]} \texttt{maxS}(\texttt{A}(\{x\}, x, 0))$$

which is an instance of the more general equality:

$$\texttt{maxS}(\texttt{A}(S, x, n)) =_{[\![]\!]} \texttt{maxS}(\texttt{A}(S \cup \{x\}, x, n), \texttt{B}(S, n)) \qquad \boxed{\mathcal{S}^{-}_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}}$$

when $x \notin S$.

The true normal form

- However, we still have the following equality:

$$\mathtt{maxS}(\mathtt{A}(\{\}, x, 0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{x\}, x, 0))$$

  which is an instance of the more general equality:

$$\mathtt{maxS}(\mathtt{A}(S, x, n)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(S \cup \{x\}, x, n), \mathtt{B}(S, n)) \qquad \mathcal{S}^-_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}$$

  when $x \notin S$.

- Applying this last equality leads us to our final subterm grammar:

$$u := \mathtt{A}(\{x\} \cup \{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n + 1).$$

  We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}$.

The true normal form

- However, we still have the following equality:

$$\mathtt{maxS}(\mathtt{A}(\{\}, x, 0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{x\}, x, 0))$$

which is an instance of the more general equality:

$$\mathtt{maxS}(\mathtt{A}(S, x, n)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(S \cup \{x\}, x, n), \mathtt{B}(S, n)) \qquad \boxed{\mathcal{S}^{-}_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}}$$

when $x \notin S$.

- Applying this last equality leads us to our final subterm grammar:

$$u := \mathtt{A}(\{x\} \cup \{x_1, \ldots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \ldots, x_n\}, n+1).$$

We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}$.

- Thanks to the $\boxed{\mathcal{L} \to \mathcal{S}^{--}_{\mathsf{nf}}}$, $\boxed{\mathcal{S}^{--}_{\mathsf{nf}} \to \mathcal{S}^{-}_{\mathsf{nf}}}$, and $\boxed{\mathcal{S}^{-}_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}}$ equations we know that these subterms satisfy the `existence` property.

The true normal form

- However, we still have the following equality:

$$\mathtt{maxS}(\mathtt{A}(\{\}, x, 0)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(\{x\}, x, 0))$$

  which is an instance of the more general equality:

$$\mathtt{maxS}(\mathtt{A}(S, x, n)) =_{[\![]\!]} \mathtt{maxS}(\mathtt{A}(S \cup \{x\}, x, n), \mathtt{B}(S, n)) \qquad \mathcal{S}^{-}_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}$$

  when $x \notin S$.

- Applying this last equality leads us to our final subterm grammar:

$$u := \mathtt{A}(\{x\} \cup \{x_1, \dots, x_n\}, x, n) \mid \mathtt{B}(\{x_1, \dots, x_n\}, n + 1).$$

  We denote this set of subterms by $\mathcal{S}_{\mathsf{nf}}$.

- Thanks to the $\mathcal{L} \to \mathcal{S}^{--}_{\mathsf{nf}}$, $\mathcal{S}^{--}_{\mathsf{nf}} \to \mathcal{S}^{-}_{\mathsf{nf}}$, and $\mathcal{S}^{-}_{\mathsf{nf}} \to \mathcal{S}_{\mathsf{nf}}$ equations we know that these subterms satisfy the `existence` property.
- However, do they also satisfy `uniqueness` and `attainability`?

Proving uniqueness

- Recall the `uniqueness` property:

$$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_m\}$$

  when $\{u_1,\ \ldots,\ u_n\}$ and $\{v_1,\ \ldots,\ v_m\}$ are incomparable.

Proving uniqueness

- Recall the `uniqueness` property:

$$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_m\}$$

  when $\{u_1,\ \ldots,\ u_n\}$ and $\{v_1,\ \ldots,\ v_m\}$ are incomparable.
- In fact, our normal form is now sufficient to prove this!

Proving uniqueness

- Recall the `uniqueness` property:

$$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_m\}$$

  when $\{u_1,\ \ldots,\ u_n\}$ and $\{v_1,\ \ldots,\ v_m\}$ are incomparable.
- In fact, our normal form is now sufficient to prove this!
- We use the following lemma:

Lemma (Independence)

Let $u \in \mathcal{S}_{nf}$ and $t = \texttt{maxS}(v_1, \ldots, v_n)$ with $\{v_1,\ \ldots,\ v_n\}$ incomparable. Then, $u \leq_{[\![]\!]} t$ if and only if there exists an i such that $u \leq_{[\![]\!]} v_i$.

Proving uniqueness

- Recall the `uniqueness` property:

$$\mathtt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \mathtt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_m\}$$

  when $\{u_1,\ \ldots,\ u_n\}$ and $\{v_1,\ \ldots,\ v_m\}$ are incomparable.
- In fact, our normal form is now sufficient to prove this!
- We use the following lemma:

Lemma (Independence)

Let $u \in \mathcal{S}_{nf}$ and $t = \mathtt{maxS}(v_1, \ldots, v_n)$ with $\{v_1,\ \ldots,\ v_n\}$ incomparable. Then, $u \leq_{[\![]\!]} t$ if and only if there exists an i such that $u \leq_{[\![]\!]} v_i$.

Proof sketch: consider the $u = \mathtt{A}(S, x, n)$, $u = \mathtt{B}(S, x)$ cases in turn and proceed by contradiction, assuming $u \not\leq_{[\![]\!]} v_i$ for all i and prove $u \not\leq_{[\![]\!]} v$, i.e. construct a σ such that $[\![u]\!]_\sigma > [\![v_i]\!]_\sigma$ for all i.

Proving uniqueness

- We now prove the uniqueness property:

Theorem (uniqueness)

For all incomparable $\{u_1,\ \ldots,\ u_n\}$ *and* $\{v_1,\ \ldots,\ v_m\}$ *in* $\mathcal{S}_{nf}$,

$$\mathtt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \mathtt{maxS}(v_1, \ldots, v_m) \iff \{u_1,\ \ldots,\ u_n\} = \{v_1,\ \ldots,\ v_n\}.$$

Proving uniqueness

- We now prove the uniqueness property:

Theorem (uniqueness)

For all incomparable $\{u_1, \ldots, u_n\}$ *and* $\{v_1, \ldots, v_m\}$ *in* $\mathcal{S}_{nf}$,

$$\mathtt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \mathtt{maxS}(v_1, \ldots, v_m) \iff \{u_1, \ldots, u_n\} = \{v_1, \ldots, v_n\}.$$

Proof.

- WTS that for any i, there exists a j such that $u_i = v_j$ (and vice versa).

□

Proving uniqueness

- We now prove the uniqueness property:

Theorem (`uniqueness`)

For all incomparable $\{u_1, \ldots, u_n\}$ *and* $\{v_1, \ldots, v_m\}$ *in* $\mathcal{S}_{nf}$*,*

$$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1, \ldots, u_n\} = \{v_1, \ldots, v_n\}.$$

Proof.

- WTS that for any i, there exists a j such that $u_i = v_j$ (and vice versa).
- For any u_i, we know that $u_i \leq_{[\![]\!]} u \leq_{[\![]\!]} v$, so by the independence lemma $u_i \leq_{[\![]\!]} v_j$ for some j. Similarly, $v_j \leq_{[\![]\!]} u_k$ for some k, so $u_i \leq_{[\![]\!]} u_k$.

$\square$

Proving uniqueness

- We now prove the uniqueness property:

Theorem (uniqueness)

For all incomparable $\{u_1, \ldots, u_n\}$ *and* $\{v_1, \ldots, v_m\}$ *in* $\mathcal{S}_{nf}$,

$$\texttt{maxS}(u_1, \ldots, u_n) =_{[\![]\!]} \texttt{maxS}(v_1, \ldots, v_m) \iff \{u_1, \ldots, u_n\} = \{v_1, \ldots, v_n\}.$$

Proof.

- WTS that for any i, there exists a j such that $u_i = v_j$ (and vice versa).
- For any u_i, we know that $u_i \leq_{[\![]\!]} u \leq_{[\![]\!]} v$, so by the independence lemma $u_i \leq_{[\![]\!]} v_j$ for some j. Similarly, $v_j \leq_{[\![]\!]} u_k$ for some k, so $u_i \leq_{[\![]\!]} u_k$.
- Because the $u_1, \ldots, u_n$ are incomparable, we know that $i = k$, which implies $v_j =_{[\![]\!]} u_i$, and (by another lemma) this implies $v_j = u_i$.
- The proof starting from v_j is identical.

□

Comparing subterms

- We have the following simple tests for semantic inequality on $\mathcal{S}_{\mathsf{nf}}$:

$$\begin{aligned}
\mathtt{A}(S,x,n) \leq_{[\![]\!]} \mathtt{A}(T,y,m) &\iff S \subseteq T \wedge x = y \wedge n \leq m \\
\mathtt{B}(S,n) \leq_{[\![]\!]} \mathtt{B}(T,m) &\iff S \subseteq T \wedge n \leq m \\
\mathtt{B}(S,n) \leq_{[\![]\!]} \mathtt{A}(T,x,m) &\iff (S \subseteq T \wedge n \leq m+1) \vee n = 0,
\end{aligned}$$

all of which are easily implementable with a confluent rewrite system.

Comparing subterms

- We have the following simple tests for semantic inequality on $\mathcal{S}_{\mathsf{nf}}$:

$$\begin{aligned}
\mathtt{A}(S,x,n) \leq_{[\![]\!]} \mathtt{A}(T,y,m) &\iff S \subseteq T \wedge x = y \wedge n \leq m \\
\mathtt{B}(S,n) \leq_{[\![]\!]} \mathtt{B}(T,m) &\iff S \subseteq T \wedge n \leq m \\
\mathtt{B}(S,n) \leq_{[\![]\!]} \mathtt{A}(T,x,m) &\iff (S \subseteq T \wedge n \leq m+1) \vee n = 0,
\end{aligned}$$

  all of which are easily implementable with a confluent rewrite system.
- Note also that $\mathtt{A}(T,x,m) \not\leq_{[\![]\!]} \mathtt{B}(S,n)$, so this covers all possible cases of $u \leq_{[\![]\!]} v$, and we thus achieve `attainability` of the normal form.

Handling Universe Cumulativity

Cumulativity

- A subtyping relation.
- Implicit in Coq.
- Implicit (but optional) in Agda.

$$\mathbb{N} \in \mathsf{U}_0 \subset \mathsf{U}_1 \cdots \subset \mathsf{U}_i \cdots$$

Cumulativity

- A subtyping relation.
- Implicit in Coq.
- Implicit (but optional) in Agda.

$$\mathbb{N} \in \mathsf{U}_0 \subset \mathsf{U}_1 \cdots \subset \mathsf{U}_i \cdots$$

Broke type uniqueness!

Make it explicit

Assaf 2014 System with explictit subtyping

- A lift function $\uparrow_i : \mathsf{U}_i \to \mathsf{U}_{i+1}$.
- $\mathsf{El}_{i+1}(\uparrow_i A) \longrightarrow \mathsf{El}_i A$
- Equivalent to implicit system.

But...

- Confluence?
- Compatibility with universe polymorphism?

The main problem

$$\frac{\Gamma \vdash A\colon \mathrm{Type}_i \qquad \Gamma, x\colon A \vdash A\colon \mathrm{Type}_j}{\Gamma \vdash \Pi x\colon A \cdot B\colon \mathrm{Type}_{\mathtt{i}(i,j)}}$$

Many way to write the same term!

$$\uparrow_1(\mathbb{N} \to \mathbb{N}) \;\equiv\; \uparrow_1\mathbb{N} \to \uparrow_1\mathbb{N} \;\equiv\; \uparrow_1\mathbb{N} \to \mathbb{N} \;\equiv\; \mathbb{N} \to \uparrow_1\mathbb{N}$$

Coq example

```
Definition cast (A: Type) := A.
Definition prod (A B: Type) := A -> B.

(* nat -> nat as Type instead of Set *)
Goal (prod nat nat) = (nat -> (cast nat)).
Proof.
now cbv.
Qed.
```

My proposal

- Choose a representative for each types.
- Restrict the syntax.

My proposal

- Choose a representative for each types.
- Restrict the syntax.

Cast of minimal/main types as representative.

$\uparrow_1(\mathbb{N} \to \mathbb{N})$ is the representative of the previous type.

The syntax

Minimal types	$M \coloneqq x \mid \mathsf{u}_i \mid \pi_{i,j}\ M\ M \mid \mathrm{Unbox}_i\, C$
Usable types	$C \coloneqq \mathrm{Box}_i\, M \mid \uparrow_i^k C$
Terms	$N \coloneqq x \mid N N \mid \lambda x : T \cdot N \mid C$
Types	$T \coloneqq \mathsf{U}_i \mid \mathsf{U}'_i \mid \mathsf{El}_i C \mid \mathsf{El}'_i M \mid \Pi x : T \cdot T$

$$\mathsf{El}_k\left(\uparrow_i^k C\right) \longrightarrow \mathsf{El}_i C$$
$$\mathsf{El}_i\left(\mathrm{Box}_i\, M\right) \longrightarrow \mathsf{El}'_i M$$

Translate the creation of a product

Translate $f\colon (A\colon \operatorname{Type}_i) := A \to A$?

$[\![A]\!]$ is a usable type. Then, the procedure is the following.

- Unbox the translation.

$$\mathrm{Unbox}_i\,[\![A]\!]$$

Translate the creation of a product

Translate $f\colon (A\colon \text{Type}_i) := A \to A$?

$[\![A]\!]$ is a usable type. Then, the procedure is the following.

- Unbox the translation.
- Create the product with the minimal type.

$$\pi_{\mathtt{i}(?,?)}\ \text{Unbox}_i\ [\![A]\!]\quad \text{Unbox}_i\ [\![A]\!]$$

Translate the creation of a product

Translate $f\colon (A\colon \text{Type}_i) := A \to A$?

$[\![A]\!]$ is a usable type. Then, the procedure is the following.

- Unbox the translation.
- Create the product with the minimal type.
- Box the result.

$$\text{Box}_{?}\left(\pi_{\mathtt{i}(?,?)}\,\text{Unbox}_i\,[\![A]\!]\;\;\text{Unbox}_i\,[\![A]\!]\right)$$

Translate the creation of a product

Translate $f\colon (A\colon \mathrm{Type}_i) := A \to A$?

$[\![A]\!]$ is a usable type. Then, the procedure is the following.

- Unbox the translation.
- Create the product with the minimal type.
- Box the result.
- Lift it.

$$\uparrow_{?}{}^{i} \left[\mathrm{Box}_{?} \left(\pi_{\mathtt{i}(?,?)}\ \mathrm{Unbox}_i\, [\![A]\!]\ \ \mathrm{Unbox}_i\, [\![A]\!]\right)\right]$$

Translate the creation of a product

Translate $f\colon (A\colon\ \mathrm{Type}_i) := A \to A$?

$[\![A]\!]$ is a usable type. Then, the procedure is the following.

- Unbox the translation.
- Create the product with the minimal type.
- Box the result.
- Lift it.

$$\uparrow_{?}{}^{i} \left[\mathrm{Box}_{?} \left(\pi_{\mathtt{i}(?,?)}\, \mathrm{Unbox}_i\, [\![A]\!] \;\; \mathrm{Unbox}_i\, [\![A]\!]\right)\right]$$

Translate the creation of a product

Translate $f\colon (A\colon \mathrm{Type}_i) := A \to A$?

$[\![A]\!]$ is a usable type. Then, the procedure is the following.

- Unbox the translation.
- Create the product with the minimal type.
- Box the result.
- Lift it.

$$\uparrow_{?}{}^{i}\left[\mathrm{Box}_{?}\left(\pi_{\mathtt{i}(?,?)}\ \mathrm{Unbox}_i\,[\![A]\!]\ \ \mathrm{Unbox}_i\,[\![A]\!]\right)\right]$$

A way to get the sort of the minimal type!