Impredicativity, Cumulativity and Product Covariance in the Logical Framework Dedukti

Thiago Felicissimo, Théo Winterhalter

FSCD 2024

July 11

Dedukti

Nowadays, growing number of proof systems being proposed and implemented

Logical frameworks (LFs) allow definition of logics in common setting

Dedukti

Nowadays, growing number of proof systems being proposed and implemented

Logical frameworks (LFs) allow definition of logics in common setting

Dedukti $\lambda\Pi$-calculus (Edinburgh LF) extended with rewrite rules

Dedukti

Nowadays, growing number of proof systems being proposed and implemented

Logical frameworks (LFs) allow definition of logics in common setting

Dedukti $\lambda\Pi$-calculus (Edinburgh LF) extended with rewrite rules

Generic proof-checker An encoding of $\mathcal{L}$ in Dedukti $\rightsquigarrow$ a proof-checker for $\mathcal{L}$

Allows cross-checking of proofs coming from proof assistants, ATPs, etc

Dedukti

Nowadays, growing number of proof systems being proposed and implemented

Logical frameworks (LFs) allow definition of logics in common setting

Dedukti $\lambda\Pi$-calculus (Edinburgh LF) extended with rewrite rules

Generic proof-checker An encoding of $\mathcal{L}$ in Dedukti $\rightsquigarrow$ a proof-checker for $\mathcal{L}$

Allows cross-checking of proofs coming from proof assistants, ATPs, etc

A framework for sharing proofs

- Fermat's Little Theorem in Matita $\rightsquigarrow$ in Coq, PVS, Lean (Thiré 2018)
- Euclids's first book in Coq $\rightsquigarrow$ in Matita, PVS, Lean (Géran 2021)
- Betrand's "Postulate" in Matita $\rightsquigarrow$ in Agda (Felicissimo *et al.* 2023)
- etc

The cumulative calculus of constructions (CC)

The underlying *cumulative type system* of the proof assistant Coq

The cumulative calculus of constructions (CC)

The underlying *cumulative type system* of the proof assistant Coq

Combines three important features:

The cumulative calculus of constructions (CC)

The underlying *cumulative type system* of the proof assistant Coq

Combines three important features:

Impredicativity

$$\Gamma, x : A \vdash_{\mathrm{CC}} P \ : \ \mathsf{Prop} \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{CC}} \Pi x : A.P \ : \ \mathsf{Prop} \qquad \text{for all } A, \Gamma$$

The cumulative calculus of constructions (CC)

The underlying *cumulative type system* of the proof assistant Coq

Combines three important features:

Impredicativity

$$\Gamma, x : A \vdash_{\mathrm{cc}} P \ : \ \mathsf{Prop} \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} \Pi x : A.P \ : \ \mathsf{Prop} \qquad \text{for all } A, \Gamma$$

Cumulativity

$$\Gamma \vdash_{\mathrm{cc}} A \ : \ \mathsf{Type}_0 \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} A \ : \ \mathsf{Type}_1$$

The cumulative calculus of constructions (CC)

The underlying *cumulative type system* of the proof assistant Coq

Combines three important features:

Impredicativity

$$\Gamma, x : A \vdash_{\mathrm{cc}} P \ : \ \mathsf{Prop} \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} \Pi x : A.P \ : \ \mathsf{Prop} \qquad \text{for all } A, \Gamma$$

Cumulativity

$$\Gamma \vdash_{\mathrm{cc}} A \ : \ \mathsf{Type}_0 \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} A \ : \ \mathsf{Type}_1$$

Product covariance

$$\Gamma \vdash_{\mathrm{cc}} f \ : \ \mathsf{Nat} \to \mathsf{Type}_0 \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} f \ : \ \mathsf{Nat} \to \mathsf{Type}_1$$

The cumulative calculus of constructions (CC)

The underlying *cumulative type system* of the proof assistant Coq

Combines three important features:

Impredicativity

$$\Gamma, x : A \vdash_{\mathrm{cc}} P \ : \ \mathsf{Prop} \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} \Pi x : A.P \ : \ \mathsf{Prop} \qquad \text{for all } A, \Gamma$$

Cumulativity

$$\Gamma \vdash_{\mathrm{cc}} A \ : \ \mathsf{Type}_0 \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} A \ : \ \mathsf{Type}_1$$

Product covariance

$$\Gamma \vdash_{\mathrm{cc}} f \ : \ \mathsf{Nat} \to \mathsf{Type}_0 \qquad \rightsquigarrow \qquad \Gamma \vdash_{\mathrm{cc}} f \ : \ \mathsf{Nat} \to \mathsf{Type}_1$$

Notation From now on, I write U_0 for Prop and U_{n+1} for Type_n

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+
Soundness				

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.

Soundness $\Gamma \vdash_{\text{CC}} t : A$ implies $[\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$ in Dedukti

Where $[\![-]\!] : \text{CC} \to \text{Dedukti}$ is the translation function

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+
Soundness	✗†	✗†	✗†	✓

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.
†: The translation function is ill-defined (see the discussion in Section 9 of the article).

Soundness $\Gamma \vdash_{\text{CC}} t : A$ implies $[\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$ in Dedukti

Where $[\![-]\!] : \text{CC} \to \text{Dedukti}$ is the translation function

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+
Soundness	✗†	✗†	✗†	✓
Confluence				

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.
†: The translation function is ill-defined (see the discussion in Section 9 of the article).

Soundness $\Gamma \vdash_{\mathrm{CC}} t : A$ implies $[\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$ in Dedukti
Where $[\![-]\!] : \mathrm{CC} \to$ Dedukti is the translation function

Confluence (in the encoding) $t_3 \ {}^*\!\longleftarrow t_1 \longrightarrow^* t_2$ implies $t_3 \longrightarrow^* \circ \ {}^*\!\longleftarrow t_2$

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+
Soundness	✗†	✗†	✗†	✓
Confluence	✗	✓‡	✗	✗

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.
†: The translation function is ill-defined (see the discussion in Section 9 of the article).
‡: Requires matching modulo associativity-commutativity, less efficient and harder to implement.

Soundness $\Gamma \vdash_{\mathrm{CC}} t : A$ implies $[\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$ in Dedukti

Where $[\![-]\!] : \mathrm{CC} \to$ Dedukti is the translation function

Confluence (in the encoding) $t_3 \ {}^*\!\longleftarrow t_1 \longrightarrow^* t_2$ implies $t_3 \longrightarrow^* \circ \ {}^*\!\longleftarrow t_2$

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+
Soundness	✗†	✗†	✗†	✓
Confluence	✗	✓‡	✗	✗
Conservativity				

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.
†: The translation function is ill-defined (see the discussion in Section 9 of the article).
‡: Requires matching modulo associativity-commutativity, less efficient and harder to implement.

Soundness $\Gamma \vdash_{\text{CC}} t : A$ implies $[\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$ in Dedukti
Where $[\![-]\!] : \text{CC} \to$ Dedukti is the translation function

Confluence (in the encoding) $t_3 \ {}^*\!\longleftarrow t_1 \longrightarrow^* t_2$ implies $t_3 \longrightarrow^* \circ \ {}^*\!\longleftarrow t_2$

Conservativity $[\![\Gamma]\!] \vdash t : [\![A]\!]$ in Dedukti implies $\Gamma \vdash_{\text{CC}} t' : A$ for some t'

Encoding CC in Dedukti?

	Assaf*	Assaf *et al.*	Thiré*	Férey+
Soundness	✗†	✗†	✗†	✓
Confluence	✗	✓‡	✗	✗
Conservativity	✗	✗	✗	✗

∗: Also handles other cumulative type systems. +: Also supports universe polymorphism.
†: The translation function is ill-defined (see the discussion in Section 9 of the article).
‡: Requires matching modulo associativity-commutativity, less efficient and harder to implement.

Soundness $\Gamma \vdash_{\mathrm{CC}} t : A$ implies $[\![\Gamma]\!] \vdash [\![t]\!] : [\![A]\!]$ in Dedukti
Where $[\![-]\!] : \mathrm{CC} \to$ Dedukti is the translation function

Confluence (in the encoding) $t_3 \ {}^*\!\longleftarrow t_1 \longrightarrow^* t_2$ implies $t_3 \longrightarrow^* \circ \ {}^*\!\longleftarrow t_2$

Conservativity $[\![\Gamma]\!] \vdash t : [\![A]\!]$ in Dedukti implies $\Gamma \vdash_{\mathrm{CC}} t' : A$ for some t'

This work

	This work
Soundness	✓
Confluence	✓
Conservativity	✓*

$\star$: In a restricted form.

This work

	This work
Soundness	✓
Confluence	✓
Conservativity	✓*

$\star$: In a restricted form.

Confluence Combines various classical results and techniques, and automated confluence/termination checkers

This work

	This work
Soundness	✓
Confluence	✓
Conservativity	✓⋆

⋆: In a restricted form.

Confluence Combines various classical results and techniques, and automated confluence/termination checkers

Soundness Fixes aforementioned problems by adopting different technique
Soundness is stated and proved in terms of inverse translation function

The Theory $\mathbb{T}_{\mathbf{CC}}$

The basis of the encoding: universes

$$\mathbb{N} : \mathbf{Type}$$
$$0 : \mathbb{N}$$
$$S : \mathbb{N} \to \mathbb{N}$$

$$U : (l : \mathbb{N}) \to \mathbf{Type}$$

Let $\underline{n} := S^n\ 0$. Then U_0 (Prop) represented by $U_{\underline{0}}$, and U_{n+1} (Type_n) represented by $U_{\underline{n+1}}$

The basis of the encoding: universes

$$\mathbb{N} : \mathbf{Type}$$
$$0 : \mathbb{N}$$
$$\mathrm{S} : \mathbb{N} \to \mathbb{N}$$

$$\mathrm{U} : (l : \mathbb{N}) \to \mathbf{Type}$$
$$\mathrm{El} : (l : \mathbb{N}) \to \mathrm{U}_l \to \mathbf{Type}$$

Let $\underline{n} := \mathrm{S}^n\ 0$. Then U_0 (Prop) represented by $\mathrm{U}_{\underline{0}}$, and U_{n+1} (Type_n) represented by $\mathrm{U}_{\underline{n+1}}$
Terms A of type $\mathrm{U}_{\underline{n}}$ are *codes* for CC types, whose *elements* are given by $\mathrm{El}_{\underline{n}}\ A$

The basis of the encoding: universes

$$\mathbb{N} : \mathbf{Type}$$
$$0 : \mathbb{N}$$
$$\mathrm{S} : \mathbb{N} \to \mathbb{N}$$

$$\mathrm{U} : (l : \mathbb{N}) \to \mathbf{Type}$$
$$\mathrm{El} : (l : \mathbb{N}) \to \mathrm{U}_l \to \mathbf{Type}$$

Let $\underline{n} := \mathrm{S}^n\ 0$. Then U_0 (Prop) represented by $\mathrm{U}_{\underline{0}}$, and U_{n+1} (Type_n) represented by $\mathrm{U}_{\underline{n+1}}$
Terms A of type $\mathrm{U}_{\underline{n}}$ are *codes* for CC types, whose *elements* are given by $\mathrm{El}_{\underline{n}}\ A$

We then add codes for dependent functions and universes themselves

The basis of the encoding: universes

$$\mathbb{N} : \mathbf{Type}$$
$$0 : \mathbb{N}$$
$$\mathrm{S} : \mathbb{N} \to \mathbb{N}$$

$$\mathrm{U} : (l : \mathbb{N}) \to \mathbf{Type}$$
$$\mathrm{El} : (l : \mathbb{N}) \to \mathrm{U}_l \to \mathbf{Type}$$

Let $\underline{n} := \mathrm{S}^n\ 0$. Then U_0 (Prop) represented by $\mathrm{U}_{\underline{0}}$, and U_{n+1} (Type_n) represented by $\mathrm{U}_{\underline{n+1}}$
Terms A of type $\mathrm{U}_{\underline{n}}$ are *codes* for CC types, whose *elements* are given by $\mathrm{El}_{\underline{n}}\ A$

We then add codes for dependent functions and universes themselves

$$\mathfrak{A} : \mathbb{N} \to \mathbb{N}$$
$$\ldots$$
$$\mathrm{u} : (l : \mathbb{N}) \to \mathrm{U}_{(\mathfrak{A}\ l)}$$
$$\mathrm{El}_{(\_)}\ \mathrm{u}_1 \longmapsto \mathrm{U}_1$$

The basis of the encoding: universes

$$\mathbb{N} : \mathbf{Type} \qquad \mathrm{U} : (l : \mathbb{N}) \to \mathbf{Type}$$
$$0 : \mathbb{N} \qquad \mathrm{El} : (l : \mathbb{N}) \to \mathrm{U}_l \to \mathbf{Type}$$
$$\mathrm{S} : \mathbb{N} \to \mathbb{N}$$

Let $\underline{n} := \mathrm{S}^n\ 0$. Then U_0 (Prop) represented by $\mathrm{U}_{\underline{0}}$, and U_{n+1} (Type_n) represented by $\mathrm{U}_{\underline{n+1}}$
Terms A of type $\mathrm{U}_{\underline{n}}$ are *codes* for CC types, whose *elements* are given by $\mathrm{El}_{\underline{n}}\ A$

We then add codes for dependent functions and universes themselves

$$\mathfrak{A} : \mathbb{N} \to \mathbb{N} \qquad \mathfrak{R} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$$
$$\ldots \qquad \ldots$$
$$\mathrm{u} : (l : \mathbb{N}) \to \mathrm{U}_{(\mathfrak{A}\ l)} \qquad \pi : (l_1\ l_2 : \mathbb{N}) \to (A : \mathrm{U}_{l_1}) \to (B : \mathrm{El}_{l_1}\ A \to \mathrm{U}_{l_2}) \to \mathrm{U}_{(\mathfrak{R}\ l_1\ l_2)}$$
$$\mathrm{El}_{(\_)}\ \mathrm{u}_1 \longmapsto \mathrm{U}_1 \qquad \mathrm{El}_{(\_)}\ (\pi_{1_1,1_2}\ \mathsf{A}\ \lambda x.\mathsf{B}\{x\}) \longmapsto (x : \mathrm{El}_{1_1}\ \mathsf{A}) \to \mathrm{El}_{1_2}\ \mathsf{B}\{x\}$$

Cumulativity

Cumulativity

Cumulativity subtype relation $\subset$ Defined by

1. "Simple" cumulativity: $\mathsf{U}_n \subset \mathsf{U}_m$ when $n \leq m$
2. Product covariance: If $A \subset B$ then $\Pi x : C.A \subset \Pi x : C.B$

Cumulativity

Cumulativity subtype relation $\subset$ Defined by

1. "Simple" cumulativity: $\mathsf{U}_n \subset \mathsf{U}_m$ when $n \leq m$
2. Product covariance: If $A \subset B$ then $\Pi x : C.A \subset \Pi x : C.B$

Therefore, it suffices to add a *lift* to coerce types from $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$ to $\Pi(\vec{x} : \vec{C}).\mathsf{U}_{n+1}$

Cumulativity

Cumulativity subtype relation $\subset$ Defined by

1. "Simple" cumulativity: $\mathsf{U}_n \subset \mathsf{U}_m$ when $n \leq m$
2. Product covariance: If $A \subset B$ then $\Pi x : C.A \subset \Pi x : C.B$

Therefore, it suffices to add a *lift* to coerce types from $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$ to $\Pi(\vec{x} : \vec{C}).\mathsf{U}_{n+1}$

We first need an internal representation in Dedukti of CC types $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$

Cumulativity

Cumulativity subtype relation $\subset$ Defined by

1. "Simple" cumulativity: $\mathsf{U}_n \subset \mathsf{U}_m$ when $n \leq m$
2. Product covariance: If $A \subset B$ then $\Pi x : C.A \subset \Pi x : C.B$

Therefore, it suffices to add a *lift* to coerce types from $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$ to $\Pi(\vec{x} : \vec{C}).\mathsf{U}_{n+1}$

We first need an internal representation in Dedukti of CC types $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$

$\mathsf{Tele} : \mathbf{Type} \qquad \diamond : \mathsf{Tele}$

$\triangleleft : (l : \mathbb{N}) \to (A : \mathsf{U}_l) \to$

$\quad (\mathsf{El}_l\ A \to \mathsf{Tele}) \to \mathsf{Tele}$

Cumulativity

Cumulativity subtype relation $\subset$ Defined by

1. "Simple" cumulativity: $\mathsf{U}_n \subset \mathsf{U}_m$ when $n \leq m$
2. Product covariance: If $A \subset B$ then $\Pi x : C.A \subset \Pi x : C.B$

Therefore, it suffices to add a *lift* to coerce types from $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$ to $\Pi(\vec{x} : \vec{C}).\mathsf{U}_{n+1}$

We first need an internal representation in Dedukti of CC types $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$

$\mathsf{Tele} : \mathbf{Type}$ $\quad \diamond : \mathsf{Tele}$ $\quad \Rightarrow : \mathsf{Tele} \to \mathbb{N} \to \mathbf{Type}$

$\triangleleft : (l : \mathbb{N}) \to (A : \mathsf{U}_l) \to$ $\quad \diamond \Rightarrow \mathtt{l}_1 \longmapsto \mathsf{U}_{\mathtt{l}_1}$

$\quad (\mathsf{El}_l\ A \to \mathsf{Tele}) \to \mathsf{Tele}$ $\quad (\mathtt{A}\ {}_{\mathtt{l}_2}\triangleleft \lambda x.\mathtt{D}\{x\}) \Rightarrow \mathtt{l}_1 \longmapsto (x : \mathsf{El}_{\mathtt{l}_2}\ \mathtt{A}) \to \mathtt{D}\{x\} \Rightarrow \mathtt{l}_1$

Cumulativity

Cumulativity subtype relation $\subset$ Defined by

1. "Simple" cumulativity: $\mathsf{U}_n \subset \mathsf{U}_m$ when $n \leq m$
2. Product covariance: If $A \subset B$ then $\Pi x : C.A \subset \Pi x : C.B$

Therefore, it suffices to add a *lift* to coerce types from $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$ to $\Pi(\vec{x} : \vec{C}).\mathsf{U}_{n+1}$

We first need an internal representation in Dedukti of CC types $\Pi(\vec{x} : \vec{C}).\mathsf{U}_n$

$$\mathsf{Tele} : \mathbf{Type} \qquad \diamond : \mathsf{Tele} \qquad \Rightarrow : \mathsf{Tele} \to \mathbb{N} \to \mathbf{Type}$$

$$\triangleleft : (l : \mathbb{N}) \to (A : \mathsf{U}_l) \to \qquad \diamond \Rightarrow \mathtt{l}_1 \longmapsto \mathsf{U}_{\mathtt{l}_1}$$

$$(\mathsf{El}_l\ A \to \mathsf{Tele}) \to \mathsf{Tele} \qquad (\mathtt{A}\ {}_{\mathtt{l}_2}\triangleleft \lambda x.\mathtt{D}\{x\}) \Rightarrow \mathtt{l}_1 \longmapsto (x : \mathsf{El}_{\mathtt{l}_2}\ \mathtt{A}) \to \mathtt{D}\{x\} \Rightarrow \mathtt{l}_1$$

We can then define the lift as

$$\uparrow : (l : \mathbb{N}) \to (D : \mathsf{Tele}) \to (D \Rightarrow l) \to (D \Rightarrow (\mathsf{S}\ l))$$

Assaf's full reflection equations

Cumulativity is *silent* in CC, but *explicit* in Dedukti using $\uparrow$

Assaf's full reflection equations

Cumulativity is *silent* in CC, but *explicit* in Dedukti using $\uparrow$

Need to ensure that different representations of the same CC term are convertible

Assaf's full reflection equations

Cumulativity is *silent* in CC, but *explicit* in Dedukti using $\uparrow$

Need to ensure that different representations of the same CC term are convertible

In a setting without product covariance, Assaf identified the following equations, where $\uparrow_{\underline{n}}^{m} D\ t$ is a notation for $\uparrow_{\underline{m-1}} D\ (\ldots(\uparrow_{\underline{n}} D\ t)\ldots)$ when $n \leq m$

$$\pi_{\underline{1+n,m}}\ (\uparrow_{\underline{n}} \diamond A)\ (\lambda x.B) \equiv \uparrow_{\underline{\mathfrak{R}(n,m)}}^{\mathfrak{R}(1+n,m)} \diamond\ (\pi_{\underline{n,m}}\ A\ (\lambda x.B))$$

$$\pi_{\underline{n,1+m}}\ A\ (\lambda x.\uparrow_{\underline{m}} \diamond B) \equiv \uparrow_{\underline{\mathfrak{R}(n,m)}}^{\mathfrak{R}(n,1+m)} \diamond\ (\pi_{\underline{n,m}}\ A\ (\lambda x.B))$$

Assaf's full reflection equations

Cumulativity is *silent* in CC, but *explicit* in Dedukti using $\uparrow$

Need to ensure that different representations of the same CC term are convertible

In a setting without product covariance, Assaf identified the following equations, where $\uparrow_{\underline{n}}^{\underline{m}} D\ t$ is a notation for $\uparrow_{\underline{m-1}} D\ (\ldots(\uparrow_{\underline{n}} D\ t)\ldots)$ when $n \leq m$

$$\pi_{\underline{1+n,m}}\ (\uparrow_{\underline{n}} \diamond A)\ (\lambda x.B) \equiv \uparrow_{\underline{\mathfrak{R}(n,m)}}^{\underline{\mathfrak{R}(1+n,m)}} \diamond (\pi_{\underline{n,m}}\ A\ (\lambda x.B))$$

$$\pi_{\underline{n,1+m}}\ A\ (\lambda x.\uparrow_{\underline{m}} \diamond B) \equiv \uparrow_{\underline{\mathfrak{R}(n,m)}}^{\underline{\mathfrak{R}(n,1+m)}} \diamond (\pi_{\underline{n,m}}\ A\ (\lambda x.B))$$

Problematic as (finite) rewrite rules Multi-step lift $\uparrow_{\underline{n}}^{\underline{m}}$ only a notation

How to know correct number of lifts when applying the rule?

A factorizing π

Solution We replace the constant π with the following one

$$\pi : (l_0\ l_1\ l_2 : \mathbb{N}) \to (A : \mathrm{U}_{(l_0 + l_1)}) \to (B : \mathrm{El}_{(l_0 + l_1)}\ A \to \mathrm{U}_{(l_0 + l_2)}) \to \mathrm{U}_{(\mathfrak{R}\ (l_0 + l_1)\ (l_0 + l_2))}$$

A factorizing π

Solution We replace the constant π with the following one

$$\pi : (l_0\ l_1\ l_2 : \mathbb{N}) \to (A : \mathrm{U}_{(l_0 + l_1)}) \to (B : \mathrm{El}_{(l_0 + l_1)}\ A \to \mathrm{U}_{(l_0 + l_2)}) \to \mathrm{U}_{(\Re\ (l_0 + l_1)\ (l_0 + l_2))}$$

If A in $\mathrm{U}_{\underline{n_A}}$ and B in $\mathrm{U}_{\underline{n_B}}$, common part of n_A and n_B can be *factorized* with rule:

$$\pi^{l_0}_{(\mathrm{S}\ l_1),(\mathrm{S}\ l_2)}\ \mathsf{A}\ \mathsf{B} \longmapsto \pi^{(\mathrm{S}\ l_0)}_{l_1,l_2}\ \mathsf{A}\ \mathsf{B}$$

Now, if $\pi^{n_0}_{\underline{n_1},\underline{n_2}}\ A\ \lambda x.B$ in normal form, then $n_1 = 0$ or $n_2 = 0$

A factorizing π

Solution We replace the constant π with the following one

$$\pi : (l_0\ l_1\ l_2 : \mathbb{N}) \to (A : \mathrm{U}_{(l_0 + l_1)}) \to (B : \mathrm{El}_{(l_0 + l_1)}\ A \to \mathrm{U}_{(l_0 + l_2)}) \to \mathrm{U}_{(\Re\ (l_0 + l_1)\ (l_0 + l_2))}$$

If A in $\mathrm{U}_{\underline{n_A}}$ and B in $\mathrm{U}_{\underline{n_B}}$, common part of n_A and n_B can be *factorized* with rule:

$$\pi^{l_0}_{(\mathrm{S}\ l_1),(\mathrm{S}\ l_2)}\ \mathsf{A}\ \mathsf{B} \longmapsto \pi^{(\mathrm{S}\ l_0)}_{l_1,l_2}\ \mathsf{A}\ \mathsf{B}$$

Now, if $\pi^{n_0}_{\underline{n_1},\underline{n_2}}\ A\ \lambda x.B$ in normal form, then $n_1 = 0$ or $n_2 = 0$

We can consider all possible cases, in each one we know how many lifts to insert

$$\pi^{0}_{(\mathrm{S}\ 1),0}\ (\uparrow_{\_} \diamond \mathsf{A})\ \mathsf{B} \longmapsto \pi^{0}_{1,0}\ \mathsf{A}\ \mathsf{B} \qquad (\ldots)$$

$$\pi^{l_1}_{0,(\mathrm{S}\ l_2)}\ \mathsf{A}\ (\lambda x.\uparrow_{\_} \diamond \mathsf{B}\{x\}) \longmapsto \uparrow_{(l_1 + l_2)} \diamond (\pi^{l_1}_{0,l_2}\ \mathsf{A}\ (\lambda x.\mathsf{B}\{x\})) \qquad (\ldots)$$

What about product covariance?

Previous rules simulate Assaf's equations, encoding "simple" cumulativity

Thiré & Férey With product covariance, $\uparrow$ must also commute with app and abs

What about product covariance?

Previous rules simulate Assaf's equations, encoding "simple" cumulativity

Thiré & Férey With product covariance, $\uparrow$ must also commute with app and abs

We add the following rules:

$$\uparrow_1 (\_\ (\_) \triangleleft \lambda x.\mathtt{D}\{x\})\ \lambda x.\mathtt{t}\{x\} \longmapsto \lambda x.\uparrow_1 \mathtt{D}\{x\}\ \mathtt{t}\{x\}$$

$$(\uparrow_1 (\_\ (\_) \triangleleft \lambda x.\mathtt{D}\{x\})\ \mathtt{t})\ \mathtt{u} \longmapsto \uparrow_1 \mathtt{D}\{\mathtt{u}\}\ (\mathtt{t}\ \mathtt{u})$$

What about product covariance?

Previous rules simulate Assaf's equations, encoding "simple" cumulativity

Thiré & Férey With product covariance, $\uparrow$ must also commute with app and abs

We add the following rules:

$$\uparrow_1 (\_\ (\_) \triangleleft \lambda x.\mathtt{D}\{x\})\ \lambda x.\mathtt{t}\{x\} \longmapsto \lambda x.\uparrow_1 \mathtt{D}\{x\}\ \mathtt{t}\{x\}$$
$$(\uparrow_1 (\_\ (\_) \triangleleft \lambda x.\mathtt{D}\{x\})\ \mathtt{t})\ \mathtt{u} \longmapsto \uparrow_1 \mathtt{D}\{\mathtt{u}\}\ (\mathtt{t}\ \mathtt{u})$$

Finally, codes in different universes may represent the same type, so we add:

$$\mathrm{El}_1\ (\uparrow_\_ \diamond \mathtt{A}) \longmapsto \mathrm{El}_{(\mathrm{P}\ 1)}\ \mathtt{A}$$
$$(\uparrow_\_ \diamond \mathtt{A})\ _1 \triangleleft \mathtt{D} \longmapsto \mathtt{A}\ _{(\mathrm{P}\ 1)} \triangleleft \mathtt{D}$$

Confluence and subject reduction

Confluence

By avoiding *non-left-linear rules*, we have made first step for proving confluence

Confluence

By avoiding *non-left-linear rules*, we have made first step for proving confluence

Yet, the proof of confluence still does not come for free

Confluence

By avoiding *non-left-linear rules*, we have made first step for proving confluence

Yet, the proof of confluence still does not come for free

Proof strategy We show confluence by decomposing

$$\beta\mathcal{R}_{\text{cc}} \quad = \quad \beta\mathcal{R}_1 \quad \cup \quad \mathcal{R}_2$$

Confluence

By avoiding *non-left-linear rules*, we have made first step for proving confluence

Yet, the proof of confluence still does not come for free

Proof strategy We show confluence by decomposing

$$\beta\mathcal{R}_{\text{cc}} \quad = \quad \beta\mathcal{R}_1 \quad \cup \quad \mathcal{R}_2$$

No critical pairs between $\beta\mathcal{R}_1$ and $\mathcal{R}_2$, and both are left-linear

By Van Oostrom and Van Raamsdonk's *orthogonal combination criterion*,
it suffices to show that both $\beta\mathcal{R}_1$ and $\mathcal{R}_2$ are confluent

Confluence of $\mathcal{R}_2$: the easy part

Critical pairs of $\mathcal{R}_2$ all close, verified with SOL and CSI$^{\text{ho}}$

Confluence of $\mathcal{R}_2$: the easy part

Critical pairs of $\mathcal{R}_2$ all close, verified with SOL and CSI$^{\text{ho}}$

By Nipkow's *critical pair criterion*, locally confluent

Confluence of $\mathcal{R}_2$: the easy part

Critical pairs of $\mathcal{R}_2$ all close, verified with SOL and CSI$^{\text{ho}}$

By Nipkow's *critical pair criterion*, locally confluent

$\mathcal{R}_2$ is strongly-normalizing (SN), verified with AProVE

Confluence of $\mathcal{R}_2$: the easy part

Critical pairs of $\mathcal{R}_2$ all close, verified with SOL and CSI$^{\text{ho}}$

By Nipkow's *critical pair criterion*, locally confluent

$\mathcal{R}_2$ is strongly-normalizing (SN), verified with AProVE

By Newman's Lemma, $\mathcal{R}_2$ is confluent

Confluence of $\beta\mathcal{R}_1$: the hard part

1

Confluence of $\beta\mathcal{R}_1$: the hard part

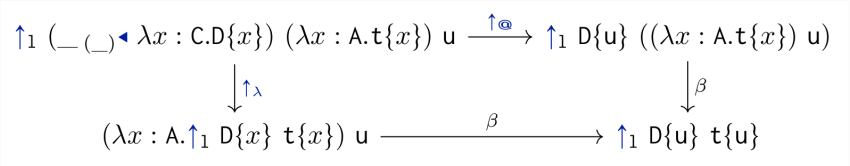
Problem 1 Cannot apply Newman's lemma because β not SN

1

Confluence of $\beta\mathcal{R}_1$: the hard part

Problem 1 Cannot apply Newman's lemma because β not SN

Problem 2 Cannot apply orthogonality criteria, because of critical pairs

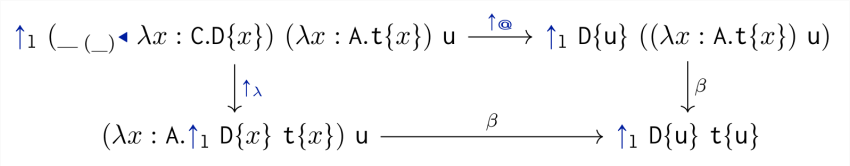


1

Confluence of $\beta\mathcal{R}_1$: the hard part

Problem 1 Cannot apply Newman's lemma because β not SN

Problem 2 Cannot apply orthogonality criteria, because of critical pairs



Solution We can still show that *orthogonal rewriting*[1] $\Longrightarrow_{\beta\mathcal{R}_1}$ satisfies diamond prop.

By the usual argument, $\Longrightarrow^*_{\beta\mathcal{R}_1} = \longrightarrow^*_{\beta\mathcal{R}_1}$, hence $\beta\mathcal{R}_1$ is confluent

[1] Also known as *developments* or *multi-step rewriting* or *simultaneous rewriting*

Subject reduction

Problem Not all rules satisfy subject reduction unconditionally

$$\pi^{(\mathsf{S}\ 1_1)}_{0,1_2}\ (\uparrow_{\_} \diamond \mathsf{A})\ \mathsf{B} \longmapsto \pi^{1_1}_{0,(\mathsf{S}\ 1_2)}\ \mathsf{A}\ \mathsf{B}$$

Subject reduction

Problem Not all rules satisfy subject reduction unconditionally

$$\pi^{(\mathsf{S}\ \mathtt{l}_1)}_{0,\mathtt{l}_2}\ (\uparrow_{\_} \diamond \mathsf{A})\ \mathsf{B} \longmapsto \pi^{\mathtt{l}_1}_{0,(\mathsf{S}\ \mathtt{l}_2)}\ \mathsf{A}\ \mathsf{B}$$

Only preserves typing if $\mathsf{S}\ (\mathtt{l}_1 \vee (\mathtt{l}_1 + \mathtt{l}_2)) \equiv \mathtt{l}_1 \vee \mathsf{S}\ (\mathtt{l}_1 + \mathtt{l}_2)$

Subject reduction

Problem Not all rules satisfy subject reduction unconditionally

$$\pi^{(\mathsf{S}\ \mathtt{l}_1)}_{0,\mathtt{l}_2}\ (\uparrow_{\_} \diamond \mathsf{A})\ \mathsf{B} \longmapsto \pi^{\mathtt{l}_1}_{0,(\mathsf{S}\ \mathtt{l}_2)}\ \mathsf{A}\ \mathsf{B}$$

Only preserves typing if $\mathsf{S}\ (\mathtt{l}_1 \vee (\mathtt{l}_1 + \mathtt{l}_2)) \equiv \mathtt{l}_1 \vee \mathsf{S}\ (\mathtt{l}_1 + \mathtt{l}_2)$

But this equation holds when $\mathtt{l}_1$ and $\mathtt{l}_2$ are replaced by concrete numbers

Subject reduction

Problem Not all rules satisfy subject reduction unconditionally

$$\pi^{(\mathsf{S}\ \mathtt{l}_1)}_{0,\mathtt{l}_2}\ (\uparrow_{\_} \diamond \mathsf{A})\ \mathsf{B} \longmapsto \pi^{\mathtt{l}_1}_{0,(\mathsf{S}\ \mathtt{l}_2)}\ \mathsf{A}\ \mathsf{B}$$

Only preserves typing if $\mathsf{S}\ (\mathtt{l}_1 \vee (\mathtt{l}_1 + \mathtt{l}_2)) \equiv \mathtt{l}_1 \vee \mathsf{S}\ (\mathtt{l}_1 + \mathtt{l}_2)$

But this equation holds when $\mathtt{l}_1$ and $\mathtt{l}_2$ are replaced by concrete numbers

Solution We can still show subject reduction for the *guarded terms* where each occurrence of π is of the form $\pi^{n_0}_{\underline{n_1},\underline{n_2}}$ for $n_0, n_1, n_2 \in \mathbb{N}$

Such terms are the only ones we need in the encoding

Soundness and Conservativity

The translation function

Problem Defining translation function $[\![-]\!] : \mathrm{CC} \to \textsc{Dedukti}$ is hard

Cumulativity is implicit in CC, so $[\![-]\!]$ needs to figure out where to insert lifts $\uparrow$

The translation function

Problem Defining translation function $[\![-]\!] : \mathrm{CC} \to \textsc{Dedukti}$ is hard

Cumulativity is implicit in CC, so $[\![-]\!]$ needs to figure out where to insert lifts $\uparrow$

Source of errors in some previous encodings

The translation function

Problem Defining translation function $[\![-]\!] : \mathrm{CC} \to \textsc{Dedukti}$ is hard

Cumulativity is implicit in CC, so $[\![-]\!]$ needs to figure out where to insert lifts $\uparrow$

Source of errors in some previous encodings

Solution We state soundness using a (partial) *inverse translation function*:

$$|-| : \textsc{Dedukti} \dashrightarrow \mathrm{CC}$$

$$|x| := x \qquad |\pi^{0}_{\underline{n},\underline{m}}\ A\ (\lambda x.B)| := \Pi x : |A|.|B|$$

$$|\mathrm{u}_{\underline{n}}| := \mathrm{U}_n \qquad |\uparrow_{\underline{n}}\ D\ t| := |t|$$

$$|\lambda x : \mathrm{El}_{\underline{n}}\ A.t| := \lambda x : |A|.|t| \qquad |t\ u| := |t|\ |u| \quad (t\ u \text{ not of previous forms})$$

The translation function

Problem Defining translation function $[\![-]\!] : \mathrm{CC} \to \textsc{Dedukti}$ is hard

Cumulativity is implicit in CC, so $[\![-]\!]$ needs to figure out where to insert lifts $\uparrow$

Source of errors in some previous encodings

Solution We state soundness using a (partial) *inverse translation function*:

$$|-| : \textsc{Dedukti} \dashrightarrow \mathrm{CC}$$

$$|x| := x \qquad |\pi^{0}_{\underline{n},\underline{m}}\ A\ (\lambda x.B)| := \Pi x : |A|.|B|$$

$$|\mathrm{u}_{\underline{n}}| := \mathrm{U}_n \qquad |\uparrow_{\underline{n}}\ D\ t| := |t|$$

$$|\lambda x : \mathrm{El}_{\underline{n}}\ A.t| := \lambda x : |A|.|t| \qquad |t\ u| := |t|\ |u| \quad (t\ u \text{ not of previous forms})$$

Define the *object terms* as the $\textsc{Dedukti}$ terms t for which $|t|$ is defined

Soundness

The main lemma:

Coherence For t_1, t_2 object terms with $\Gamma \vdash t_i : A$, if $|t_1| = |t_2|$ then $t_1 \equiv t_2$

Soundness

The main lemma:

Coherence For t_1, t_2 object terms with $\Gamma \vdash t_i : A$, if $|t_1| = |t_2|$ then $t_1 \equiv t_2$

Ensures that different representations of same CC term are convertible in Dedukti

Soundness

The main lemma:

Coherence For t_1, t_2 object terms with $\Gamma \vdash t_i : A$, if $|t_1| = |t_2|$ then $t_1 \equiv t_2$

Ensures that different representations of same CC term are convertible in Dedukti

Soundness If $\Gamma \vdash_{\mathrm{CC}} t : A$ then $\Gamma' \vdash t' : \mathrm{El}_{\underline{n}}\, A'$ for some Γ', t', A', n
with $|\Gamma'| = \Gamma$ and $|t'| = t$ and $|A'| = A$

Soundness

The main lemma:

Coherence For t_1, t_2 object terms with $\Gamma \vdash t_i : A$, if $|t_1| = |t_2|$ then $t_1 \equiv t_2$

Ensures that different representations of same CC term are convertible in Dedukti

Soundness If $\Gamma \vdash_{\text{CC}} t : A$ then $\Gamma' \vdash t' : \text{El}_{\underline{n}}\ A'$ for some Γ', t', A', n
with $|\Gamma'| = \Gamma$ and $|t'| = t$ and $|A'| = A$

The direct translation function can then be extract from the proof

Conservativity

Using inverse translation function, conservativity becomes

$$\text{If } |\Gamma| \vdash_{\text{cc}} |A| : \mathsf{U}_n \text{ and } \Gamma \vdash t : \underline{\mathsf{El}}_n\, A \text{ then } |\Gamma| \vdash_{\text{cc}} t' : |A| \text{ for some } t'$$

Conservativity

Using inverse translation function, conservativity becomes

$$\text{If } |\Gamma| \vdash_{\text{CC}} |A| : \mathsf{U}_n \text{ and } \Gamma \vdash t : \mathsf{El}_{\underline{n}}\, A \text{ then } |\Gamma| \vdash_{\text{CC}} t' : |A| \text{ for some } t'$$

Problem Not all terms t correspond directly to CC terms

Conservativity

Using inverse translation function, conservativity becomes

$$\text{If } |\Gamma| \vdash_{\text{CC}} |A| : \mathsf{U}_n \text{ and } \Gamma \vdash t : \underline{\mathsf{El}}_n\ A \text{ then } |\Gamma| \vdash_{\text{CC}} t' : |A| \text{ for some } t'$$

Problem Not all terms t correspond directly to CC terms

Solution We instead only show *object conservativity*:

$$\text{If } t \text{ is an object term and } |\Gamma| \vdash_{\text{CC}} |A| : \mathsf{U}_n \text{ and } \Gamma \vdash t : \underline{\mathsf{El}}_n\ A \text{ then } |\Gamma| \vdash_{\text{CC}} |t| : |A|$$

In practical applications, only object terms are used anyway

Conservativity

Using inverse translation function, conservativity becomes

$$\text{If } |\Gamma| \vdash_{\text{CC}} |A| : \mathsf{U}_n \text{ and } \Gamma \vdash t : \mathrm{El}_{\underline{n}}\ A \text{ then } |\Gamma| \vdash_{\text{CC}} t' : |A| \text{ for some } t'$$

Problem Not all terms t correspond directly to CC terms

Solution We instead only show *object conservativity*:

If t is an object term and $|\Gamma| \vdash_{\text{CC}} |A| : \mathsf{U}_n$ and $\Gamma \vdash t : \mathrm{El}_{\underline{n}}\ A$ then $|\Gamma| \vdash_{\text{CC}} |t| : |A|$

In practical applications, only object terms are used anyway

It should be possible to prove full conservativity (see concl. of the paper)

Conclusion

Conclusion

An encoding of CC satisfying

	This work
Soundness	✓
Confluence	✓
Conservativity	✓⋆

⋆: Only for object terms.

Conclusion

An encoding of CC satisfying

	This work
Soundness	✓
Confluence	✓
Conservativity	✓⋆

⋆: Only for object terms.

Future work Universe polymorphism and implementation

Conclusion

An encoding of CC satisfying

	This work
Soundness	✓
Confluence	✓
Conservativity	✓*

$\star$: Only for object terms.

Future work Universe polymorphism and implementation

Challenge for confluence checkers Proving confluence of $\beta\mathcal{R}_{\text{CC}}$ automatically

Conclusion

An encoding of CC satisfying

	This work
Soundness	✓
Confluence	✓
Conservativity	✓*

$\star$: Only for object terms.

Future work Universe polymorphism and implementation

Challenge for confluence checkers Proving confluence of $\beta\mathcal{R}_{\text{CC}}$ automatically

Thank you!