Adequate and Computational Encodings in the Logical Framework Dedukti

Thiago Felicissimo

Formal Structures for Computation and Deduction 2022, Haifa

August 4, 2022

Logical Frameworks & Dedukti

Logical Frameworks

Each year, more and more new type theories and systems are proposed

Hard to see how they all relate

Logical Frameworks address this heterogeneity

A common foundation for defining type theories and logics

Of theoretical interest (decomposing and comparing theories)

and practical one (prototyping developments, checking proofs)

Dedukti

Edinburgh Logical Framework (LF) Historically, very influential framework

It's just the λ-calculus with dependent types! (also known as $\lambda\Pi$-calculus)

Dependent types allow to express deduction!

But computation can only be expressed as deduction...

$$\pi_1(M, N) \hookrightarrow_{\beta_{\pi_1}} M \qquad \rightsquigarrow \qquad \beta_{\pi_1} : eq\ (\pi_1\ [\![M]\!]\ [\![N]\!])\ [\![M]\!]$$

Dedukti

Edinburgh Logical Framework (LF) Historically, very influential framework

It's just the λ-calculus with dependent types! (also known as $\lambda\Pi$-calculus)

Dependent types allow to express deduction!

But computation can only be expressed as deduction...

$$\pi_1(M, N) \hookrightarrow_{\beta_{\pi_1}} M \qquad \rightsquigarrow \qquad \beta_{\pi_1} : eq\ (\pi_1\ [\![M]\!]\ [\![N]\!])\ [\![M]\!]$$

Dedukti Extends LF with user-defined rewrite rules $\mathcal{R}$, typing modulo $\equiv_{\beta\mathcal{R}}$

$$\pi_1\ M\ N \hookrightarrow_{beta_{\pi_1}} M \qquad \in \mathcal{R}$$

Handles both building blocks of modern logics: deduction and computation

In practice, useful for rechecking and sharing proofs (see the EuroProofNet project)

What is a Dedukti encoding?

A **theory** is a pair $(\Sigma, \mathcal{R})$ where

- $\Sigma = \{c : A, d : B, ...\}$ is a signature (constant declarations with their types)
- $\mathcal{R} = \{l \hookrightarrow r, ...\}$ is a set of rewrite rules

Used to represent object logics in Dedukti: $\mathcal{O}$ represented by $D[\mathcal{O}] = (\Sigma_{\mathcal{O}}, \mathcal{R}_{\mathcal{O}})$

What is a Dedukti encoding?

A **theory** is a pair $(\Sigma, \mathcal{R})$ where

- $\Sigma = \{c : A, d : B, ...\}$ is a signature (constant declarations with their types)
- $\mathcal{R} = \{l \hookrightarrow r, ...\}$ is a set of rewrite rules

Used to represent object logics in Dedukti: $\mathcal{O}$ represented by $D[\mathcal{O}] = (\Sigma_{\mathcal{O}}, \mathcal{R}_{\mathcal{O}})$

An **encoding** of $\mathcal{O}$: a theory $D[\mathcal{O}]$ and a translation function $[\![-]\!] : \Lambda_{\mathcal{O}} \to \Lambda_{\mathtt{DK}}$

An hierarchy of encodings

An encoding is sound if:

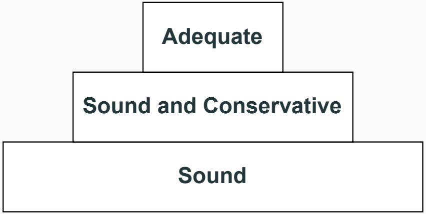
$$\vdash_{\mathcal{O}} M : A \quad \text{implies} \quad \vdash_{\mathtt{DK}} [\![M]\!] : \mathit{El}\ [\![A]\!]$$

An encoding is conservative if:

$$\vdash_{\mathtt{DK}} M : \mathit{El}\ [\![A]\!] \quad \text{implies} \quad \exists N \text{ s.t. } \vdash_{\mathcal{O}} N : A$$

An encoding is adequate if for each type A:

$$[\![-]\!] \text{ is a bijection between } A \text{ and } \mathit{El}\ [\![A]\!]$$



The problem of conservativity and adequacy

Unlike with LF encodings, Dedukti encodings proposed until now are not adequate

Actually, just showing conservativity of Dedukti encodings is already **very hard**

For many recently proposed encodings, still only a conjecture...

The problem of conservativity and adequacy

Unlike with LF encodings, Dedukti encodings proposed until now are not adequate

Actually, just showing conservativity of Dedukti encodings is already **very hard**

For many recently proposed encodings, still only a conjecture...

Where does this problem come from?

The cause can actually be traced back to the first Dedukti encoding, 2007 Cousineau & Dowek's encoding of (functional) Pure Type Systems

The cause of the problem, in a nutshell

Cousineau & Dowek's idea is to represent object functions by framework's functions

$$(\star) \quad El\ (Prod\ A\ B) \hookrightarrow \Pi x : El\ A . El\ (B\ x)$$

Problem To show conservativity, we need to assume that β terminates

But because of $(\star)$, β might not terminate (when it does, proof is non trivial)

The cause of the problem, in a nutshell

Cousineau & Dowek's idea is to represent object functions by framework's functions

$$(\star) \quad \mathit{El}\ (\mathit{Prod}\ A\ B) \longrightarrow \Pi x : \mathit{El}\ A . \mathit{El}\ (B\ x)$$

Problem To show conservativity, we need to assume that β terminates

But because of (⋆), β might not terminate (when it does, proof is non trivial)

This work A different encoding of (functional) Pure Type Systems, without (⋆), where β always normalizes

Easy conservative proof and an adequacy theorem

An encoding both adequate (like in LF) and computational (like in Dedukti)

Encoding Pure Type Systems in Dedukti

Pure Type Systems

Pure Type Systems (or PTSs) is a class of type theories with two forms of types: dependent functions ($\Pi x : A.B$) and universes (or sorts, written as $s_1, s_2, \ldots$)

$$M, N, A, B ::= x \in \mathcal{V} \mid s \in \mathcal{S} \mid \Pi x : A.B \mid \lambda x : A.M \mid MN$$

$$(\lambda x : A.M)N \hookrightarrow_\beta M\{N/x\}$$

Each PTS described by a specification $(\mathcal{S}, \mathcal{A}, \mathcal{R})$, where $\mathcal{S}$ is a set of sorts and $\mathcal{A} \subseteq \mathcal{S}^2$, $\mathcal{R} \subseteq \mathcal{S}^3$

$$\frac{}{\Gamma \vdash s_1 : s_2}(s_1, s_2) \in \mathcal{A} \qquad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.B : s_3}(s_1, s_2, s_3) \in \mathcal{R}$$

For the rest of this talk, we assume $\mathcal{A}, \mathcal{R}$ are functional

The theory $(\Sigma_{PTS}, \mathcal{R}_{PTS})$

Universes

$$
\begin{array}{rll}
\Sigma_{PTS} \ni & U_s : \textbf{Type} & \text{for } s \in \mathcal{S} \\
\Sigma_{PTS} \ni & El_s(A : U_s) : \textbf{Type} & \text{for } s \in \mathcal{S} \\
\Sigma_{PTS} \ni & u_{s_1} : U_{s_2} & \text{for } (s_1, s_2) \in \mathcal{A} \\
\mathcal{R}_{PTS} \ni & El_{s_2}\ u_{s_1} \longrightarrow U_{s_1} & \text{for } (s_1, s_2) \in \mathcal{A}
\end{array}
$$

Dependent functions (Cousineau & Dowek)

$$\Sigma_{PTS} \ni Prod_{s_1,s_2}(A : U_{s_1})(B : El_{s_1}\ A \to U_{s_2}) : U_{s_3} \qquad \text{for } (s_1, s_2, s_3) \in \mathcal{R}$$

$$\mathcal{R}_{PTS} \ni El_{s_3}\ (Prod_{s_1,s_2}\ A\ B) \longrightarrow \Pi x : El_{s_1}\ A.El_{s_2}\ (B\ x)$$

The theory $(\Sigma_{PTS}, \mathcal{R}_{PTS})$

Universes

$$\Sigma_{PTS} \ni \quad U_s : \mathbf{Type} \qquad \text{for } s \in \mathcal{S}$$

$$\Sigma_{PTS} \ni \quad El_s(A : U_s) : \mathbf{Type} \qquad \text{for } s \in \mathcal{S}$$

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

$$\mathcal{R}_{PTS} \ni \quad El_{s_2}\ u_{s_1} \longrightarrow U_{s_1} \qquad \text{for } (s_1, s_2) \in \mathcal{A}$$

Dependent functions

$$\Sigma_{PTS} \ni Prod_{s_1,s_2}(A : U_{s_1})(B : El_{s_1}\ A \to U_{s_2}) : U_{s_3} \qquad \text{for } (s_1, s_2, s_3) \in \mathcal{R}$$

$$\Sigma_{PTS} \ni abs_{s_1,s_2}(A : U_{s_1})(B : El_{s_1}\ A \to U_{s_2})(M : \Pi x : El_{s_1}\ A.El_{s_2}\ (B\ x)) : El_{s_3}(Prod_{s_1,s_2}\ A\ B)$$

$$\Sigma_{PTS} \ni app_{s_1,s_2}(A : U_{s_1})(B : El_{s_1}\ A \to U_{s_2})(M : El_{s_3}(Prod_{s_1,s_2}\ A\ B))(N : El_{s_1}\ A) : El_{s_2}(B\ N)$$

$$\mathcal{R}_{PTS} \ni app_{s_1,s_2}\ A\ B\ (abs_{s_1,s_2}\ A'\ B'\ M)\ N \longrightarrow_{beta_{s_1,s_2}} M\ N$$

Defining the translation function

$$
\begin{aligned}
[\![x]\!] &= x \\
[\![s]\!] &= u_s \\
[\![\Pi x : A.B]\!] &= \; ? \\
[\![\lambda x : A.M]\!] &= \; ? \\
[\![MN]\!] &= \; ?
\end{aligned}
$$

Defining the translation function

$$\llbracket x \rrbracket = x$$
$$\llbracket s \rrbracket = u_s$$
$$\llbracket \Pi x : A.B \rrbracket = Prod_{?,?}\ \llbracket A \rrbracket\ (\lambda x.\llbracket B \rrbracket)$$
$$\llbracket \lambda x : A.M \rrbracket = abs_{?,?}\ \llbracket A \rrbracket\ ?\ (\lambda x.\llbracket M \rrbracket)$$
$$\llbracket MN \rrbracket = app_{?,?}\ ?\ ?\ \llbracket M \rrbracket\ \llbracket N \rrbracket$$

Defining the translation function

$$\llbracket x \rrbracket_\Gamma = x$$

$$\llbracket s \rrbracket_\Gamma = u_s$$

$$\llbracket \Pi x : A.B \rrbracket_\Gamma = Prod_{s_1,s_2}\ \llbracket A \rrbracket_\Gamma\ (\lambda x.\llbracket B \rrbracket_{\Gamma,x:A}) \quad \text{where } \Gamma \vdash A : s_1 \text{ and } \Gamma, x : A \vdash B : s_2$$

$$\llbracket \lambda x : A.M \rrbracket_\Gamma = abs_{?,?}\ \llbracket A \rrbracket_\Gamma\ ?\ (\lambda x.\llbracket M \rrbracket_{\Gamma,x:A})$$

$$\llbracket MN \rrbracket_\Gamma = app_{?,?}\ ?\ ?\ \llbracket M \rrbracket_\Gamma\ \llbracket N \rrbracket_\Gamma$$

Defining the translation function

$$\llbracket x \rrbracket_\Gamma = x$$

$$\llbracket s \rrbracket_\Gamma = u_s$$

$$\llbracket \Pi x : A.B \rrbracket_\Gamma = Prod_{s_1,s_2}\ \llbracket A \rrbracket_\Gamma\ (\lambda x. \llbracket B \rrbracket_{\Gamma, x:A}) \quad \text{where } \Gamma \vdash A : s_1 \text{ and } \Gamma, x : A \vdash B : s_2$$

$$\llbracket \lambda x : A.M \rrbracket_\Gamma = abs_{?,?}\ \llbracket A \rrbracket_\Gamma\ (\lambda x. \llbracket B \rrbracket_{\Gamma, x:A})\ (\lambda x. \llbracket M \rrbracket_{\Gamma, x:A}) \quad \text{where } \Gamma, x : A \vdash M : B$$

$$\llbracket MN \rrbracket_\Gamma = app_{?,?}\ \llbracket A \rrbracket_\Gamma\ (\lambda x. \llbracket B \rrbracket_{\Gamma, x:A})\ \llbracket M \rrbracket_\Gamma\ \llbracket N \rrbracket_\Gamma \quad \text{where } \Gamma \vdash M : \Pi x : A.B$$

Defining the translation function

$$
\begin{aligned}
[\![x]\!]_\Gamma &= x \\
[\![s]\!]_\Gamma &= u_s \\
[\![\Pi x : A.B]\!]_\Gamma &= \mathit{Prod}_{s_1,s_2}\ [\![A]\!]_\Gamma\ (\lambda x.[\![B]\!]_{\Gamma,x:A}) \quad \text{where } \Gamma \vdash A : s_1 \text{ and } \Gamma, x : A \vdash B : s_2 \\
[\![\lambda x : A.M]\!]_\Gamma &= \mathit{abs}_{?,?}\ [\![A]\!]_\Gamma\ (\lambda x.[\![B]\!]_{\Gamma,x:A})\ (\lambda x.[\![M]\!]_{\Gamma,x:A}) \quad \text{where } \Gamma, x : A \vdash M : B \\
[\![MN]\!]_\Gamma &= \mathit{app}_{?,?}\ [\![A]\!]_\Gamma\ (\lambda x.[\![B]\!]_{\Gamma,x:A})\ [\![M]\!]_\Gamma\ [\![N]\!]_\Gamma \quad \text{where } \Gamma \vdash M : \Pi x : A.B
\end{aligned}
$$

Problem 1: A, B not syntactically unique, thus $[\![M]\!] \subseteq \Lambda_{\mathtt{DK}}$ instead of $[\![M]\!] \in \Lambda_{\mathtt{DK}}$

Problem 2: Not a valid structural recursion: A, B are not subterms

Defining the translation function

$$\llbracket x \rrbracket_\Gamma = x$$

$$\llbracket s \rrbracket_\Gamma = u_s$$

$$\llbracket \Pi x : A.B \rrbracket_\Gamma = Prod_{s_1,s_2}\ \llbracket A \rrbracket_\Gamma\ (\lambda x.\llbracket B \rrbracket_{\Gamma,x:A}) \quad \text{where } \Gamma \vdash A : s_1 \text{ and } \Gamma, x : A \vdash B : s_2$$

$$\llbracket \lambda x : A.M \rrbracket_\Gamma = abs_{?,?}\ \llbracket A \rrbracket_\Gamma\ (\lambda x.\llbracket B \rrbracket_{\Gamma,x:A})\ (\lambda x.\llbracket M \rrbracket_{\Gamma,x:A}) \quad \text{where } \Gamma, x : A \vdash M : B$$

$$\llbracket MN \rrbracket_\Gamma = app_{?,?}\ \llbracket A \rrbracket_\Gamma\ (\lambda x.\llbracket B \rrbracket_{\Gamma,x:A})\ \llbracket M \rrbracket_\Gamma\ \llbracket N \rrbracket_\Gamma \quad \text{where } \Gamma \vdash M : \Pi x : A.B$$

Problem 1: A, B not syntactically unique, thus $\llbracket M \rrbracket \subseteq \Lambda_{\mathtt{DK}}$ instead of $\llbracket M \rrbracket \in \Lambda_{\mathtt{DK}}$

Problem 2: Not a valid structural recursion: A, B are not subterms

Solution: Add the necessary data to the syntax

Explicitly-typed Pure Type Systems (EPTS)

$$(s_1, s_2, s_3) \in \mathcal{R} \; \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.B : s_3} \; \text{Prod}$$

$$(s_1, s_2, s_3) \in \mathcal{R} \; \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B} \; \text{Abs}$$

$$(s_1, s_2, s_3) \in \mathcal{R} \; \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad \Gamma \vdash M : \Pi x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B\{N/x\}} \; \text{App}$$

Explicitly-typed Pure Type Systems (EPTS)

$$(s_1, s_2, s_3) \in \mathcal{R} \; \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi_{s_1,s_2}(A, [x]B) : s_3} \; \text{Prod}$$

$$(s_1, s_2, s_3) \in \mathcal{R} \; \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda_{s_1,s_2}(A, [x]B, [x]M) : \Pi_{s_1,s_2}(A, [x]B)} \; \text{Abs}$$

$$(s_1, s_2, s_3) \in \mathcal{R} \; \frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad \Gamma \vdash M : \Pi_{s_1,s_2}(A, [x]B) \qquad \Gamma \vdash N : A}{\Gamma \vdash @_{s_1,s_2}(A, [x]B, M, N) : B\{N/x\}} \; \text{App}$$

Theorem $\Gamma \vdash_{PTS} M : A$ iff there are Γ', M', A' with $\Gamma' \vdash_{EPTS} M' : A'$, where Γ', M', A' are erased to Γ, M, A

Defining the translation function

$$
\begin{aligned}
[\![x]\!] &= x \\
[\![s]\!] &= u_s \\
[\![\Pi_{s_1,s_2}(A, {}_{[x]}B)]\!] &= Prod_{s_1,s_2}\ [\![A]\!]\ (\lambda x.[\![B]\!]) \\
[\![\lambda_{s_1,s_2}(A, {}_{[x]}B, {}_{[x]}M)]\!] &= abs_{s_1,s_2}\ [\![A]\!]\ (\lambda x.[\![B]\!])\ (\lambda x.[\![M]\!]) \\
[\![@_{s_1,s_2}(A, {}_{[x]}B, M, N)]\!] &= app_{s_1,s_2}\ [\![A]\!]\ (\lambda x.[\![B]\!])\ [\![M]\!]\ [\![N]\!]
\end{aligned}
$$

Very natural definition

Does not need to prove that M is typed on Γ to apply $[\![-]\!]$

Soundness, Conservativity and Adequacy

The easy bit: soundness

Before soundness, show that the **encoding is computational**

$$M \hookrightarrow N \qquad \text{implies} \qquad [\![M]\!] \hookrightarrow^{+} [\![N]\!]$$

Not satisfied by LF encodings, but here Dedukti shines!

Soundness If $\Gamma \vdash_{EPTS} M : A$ then $[\![\Gamma]\!] \vdash_{\mathtt{DK}} [\![M]\!] : El_{s_A}\ [\![A]\!]$

Simple proof by induction on the derivation

Conservativity, first try

If $[\![\Gamma]\!] \vdash_{\mathtt{DK}} M : El_{s_A}\ [\![A]\!]$ we would like to show $\Gamma \vdash_{EPTS} [\![M]\!]^{-1} : A$

Problem If M not in β-normal form, no way to inverse it. What is $[\![N]\!]^{-1}$ of

$$N = (\lambda x.(\lambda z.z(\lambda y.[\![A_2]\!]))\ (Prod_{s_1,s_2}\ x))\ [\![A_1]\!] \qquad ?$$

Conservativity, first try

If $[\![\Gamma]\!] \vdash_{\mathtt{DK}} M : El_{s_A}\ [\![A]\!]$ we would like to show $\Gamma \vdash_{EPTS} [\![M]\!]^{-1} : A$

Problem If M not in β-normal form, no way to inverse it. What is $[\![N]\!]^{-1}$ of

$$N = (\lambda x.(\lambda z.z(\lambda y.[\![A_2]\!]))\ (Prod_{s_1,s_2}\ x))\ [\![A_1]\!] \qquad ?$$

Solution Take the β-normal form!

$NF_\beta(N) = Prod_{s_1,s_2}\ [\![A_1]\!]\ (\lambda y.[\![A_2]\!])$, thus $[\![NF_\beta(N)]\!]^{-1} = \Pi_{s_1,s_2}(A_1, {}_{[y]}A_2)$

Conservativity, first try

If $[\![\Gamma]\!] \vdash_{\mathtt{DK}} M : El_{s_A}\ [\![A]\!]$ we would like to show $\Gamma \vdash_{EPTS} [\![M]\!]^{-1} : A$

Problem If M not in β-normal form, no way to inverse it. What is $[\![N]\!]^{-1}$ of

$$N = (\lambda x.(\lambda z.z(\lambda y.[\![A_2]\!]))\ (Prod_{s_1,s_2}\ x))\ [\![A_1]\!] \qquad ?$$

Solution Take the β-normal form!

$NF_\beta(N) = Prod_{s_1,s_2}\ [\![A_1]\!]\ (\lambda y.[\![A_2]\!])$, thus $[\![NF_\beta(N)]\!]^{-1} = \Pi_{s_1,s_2}(A_1, {}_{[y]}A_2)$

Solution also used in LF, where β is known to be SN (strongly normalizing)

But in Dedukti, rewrite rules extend the typing relation of the λΠ-calculus

Is β still SN in Dedukti?

A simple proof that β is SN

$\mathcal{R}$ is **arity preserving** if (roughly) no Π appears at right-hand sides of rules in $\mathcal{R}$

Theorem If $\mathcal{R}$ is arity preserving and $\beta\mathcal{R}$ is confluent, then β is SN in Dedukti

Proof by erasure into the simply-typed λ-calculus

A simple proof that β is SN

$\mathcal{R}$ is **arity preserving** if (roughly) no Π appears at right-hand sides of rules in $\mathcal{R}$

Theorem If $\mathcal{R}$ is arity preserving and $\beta\mathcal{R}$ is confluent, then β is SN in Dedukti

Proof by erasure into the simply-typed λ-calculus

Example The rewrite rule used by Cousineau & Dowek to encode PTS

$$(\star) \quad El_{s_3}\ (Prod_{s_1,s_2}\ A\ B) \longrightarrow \Pi x : El_{s_1}\ A.El_{s_2}\ (B\ x)$$

is not arity preserving

Expected, since we know $(\star)$ can break SN of β

Conservativity, finally

$\mathcal{R}_{PTS}$ is arity preserving and $\beta\mathcal{R}_{PTS}$ is confluent (can be seen as orthogonal CRS)

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

$$app_{s_1,s_2}\ A\ B\ (abs_{s_1,s_2}\ A'\ B'\ M)\ N \hookrightarrow_{beta_{s_1,s_2}} M\ N$$

Thus, to show conservativity it suffices to consider β-normal forms

Conservativity, finally

$\mathcal{R}_{PTS}$ is arity preserving and $\beta\mathcal{R}_{PTS}$ is confluent (can be seen as orthogonal CRS)

$$\begin{aligned} &El_{s_2}\ u_{s_1} \hookrightarrow U_{s_1} \\ &app_{s_1,s_2}\ A\ B\ (abs_{s_1,s_2}\ A'\ B'\ M)\ N \hookrightarrow_{beta_{s_1,s_2}} M\ N \end{aligned}$$

Thus, to show conservativity it suffices to consider β-normal forms

Theorem If $[\![\Gamma]\!] \vdash_{\mathtt{DK}} M : El_{s_A}\ [\![A]\!]$ and $\underline{M \text{ in } \beta\text{-normal form}}$ then $\Gamma \vdash_{EPTS} [\![M]\!]^{-1} : A$

Proof by induction on M, some work but not hard

Adequacy, a simple corollary

Let $\Gamma \vdash_{EPTS} A : s_A$

Theorem $[\![-]\!]$ and $[\![-]\!]^{-1}$ form a bijection $A \simeq NF_\beta(El_{s_A}\ [\![A]\!])$

Consequence of soundness and conservativity

Conclusion

Takeaway lesson

Cousineau & Dowek's encoding represents object β by Dedukti's β

SN of β becomes **property of the theory**

Conservativity needs SN of β, and thus is made dependent on the encoded system

But logical frameworks should be **agnostic** to such properties

Takeaway lesson

Cousineau & Dowek's encoding represents object β by Dedukti's β

SN of β becomes **property of the theory**

Conservativity needs SN of β, and thus is made dependent on the encoded system

But logical frameworks should be **agnostic** to such properties

In our encoding, *beta* separated from β. Here, β represents pending substitution

$$app\ (abs\ (\lambda x.M))\ N \hookrightarrow_{beta} (\lambda x.M)N \hookrightarrow_{\beta} M\{N/x\}$$

Suggests that SN of β should be a **property of the framework**

It is SN of $\beta\mathcal{R}$ which is a **property of the theory**: if we instantiate our encoding with non-normalizing PTS, $\beta\mathcal{R}$ will not be SN, but β will always be SN

Future work

What about systems that are not Pure Type Systems?

Future work Encoding can be probably extended to general definition of purely computational type theories

Future work

What about systems that are not Pure Type Systems?

Future work Encoding can be probably extended to general definition of purely computational type theories

Elephant in the room $\mathit{app}_{s_1,s_2}\, A\, (\lambda x.B)\, M\, N$ much bigger then MN, benchmarks show 16 times performance loss when checking Fermat's Little Theorem

Future work Explore extensions of Dedukti with erased (not implicit!) arguments

Improve performance, but also make $[\![-]\!]$ easy to define

Future work

What about systems that are not Pure Type Systems?

Future work Encoding can be probably extended to general definition of purely computational type theories

Elephant in the room $app_{s_1,s_2}\, A\,(\lambda x.B)\, M\, N$ much bigger then MN, benchmarks show 16 times performance loss when checking Fermat's Little Theorem

Future work Explore extensions of Dedukti with erased (not implicit!) arguments

Improve performance, but also make $[\![-]\!]$ easy to define

Thank you!