

A proof checking kernel for the $\lambda\Pi$ -calculus modulo

Mathieu Boespflug, École Polytechnique



PhD defense, 18 january 2011



Funded by



Pythia of Delphi



True

False



Proof implies truth.¹

¹ For any reasonable notion of proof.



Formal systems

Example

☛ The language of *formulae*

words

☛ The set of *axioms* (or *assumptions*)

a-z,ε

$$\text{(ax)} \frac{P \text{ is an axiom}}{P \text{ is a palindrome}}$$

☛ The language of *proofs*

$$\text{(ext)} \frac{P \text{ is a palindrome}}{xP x \text{ is a palindrome}}$$

$$\text{(concat)} \frac{P \text{ is a palindrome} \quad Q \text{ is a palindrome}}{QPQ \text{ is a palindrome}}$$

☛ *Theorems* are formulae that have proofs.



Palindromes: example

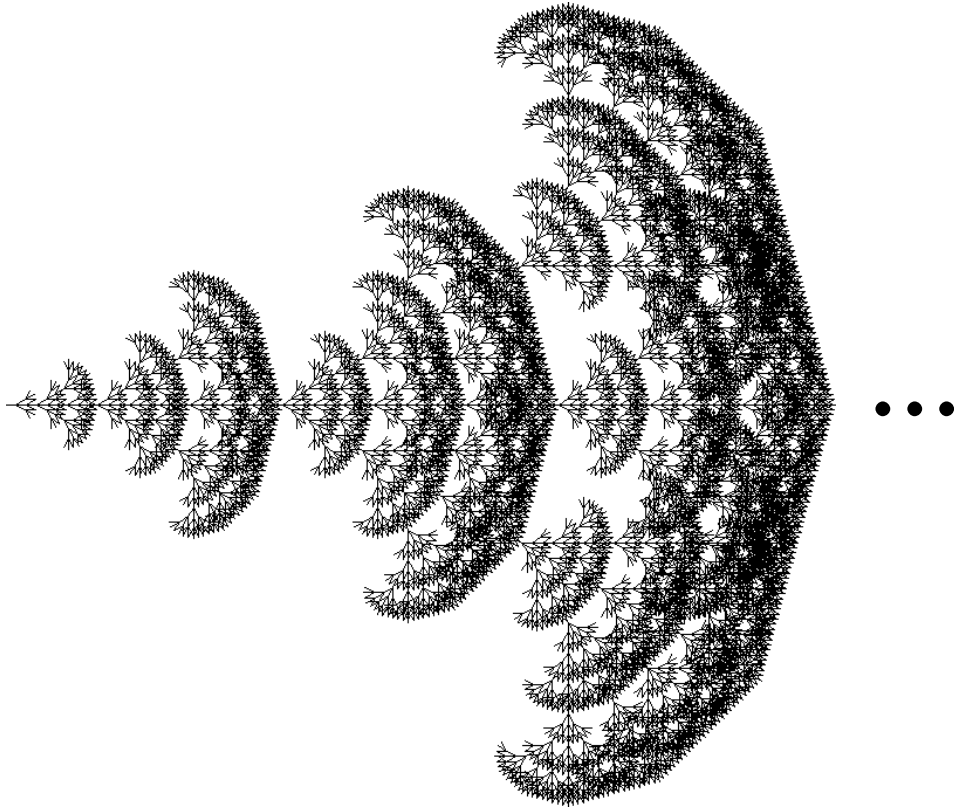
$$\begin{array}{l} \text{(ax)} \frac{\text{d is an axiom}}{\text{t is a palindrome}} \\ \text{(ext)} \frac{\text{t is a palindrome}}{\text{rtr is a palindrome}} \\ \text{(ext)} \frac{\text{rtr is a palindrome}}{\text{artra is a palindrome}} \\ \text{(ext)} \frac{\text{artra is a palindrome}}{\text{tartrat is a palindrome}} \\ \text{(ext)} \frac{\text{tartrat is a palindrome}}{\text{etartrate is a palindrome}} \\ \text{(ext)} \frac{\text{etartrate is a palindrome}}{\text{detartrated is a palindrome}} \end{array}$$


Palindromes: example

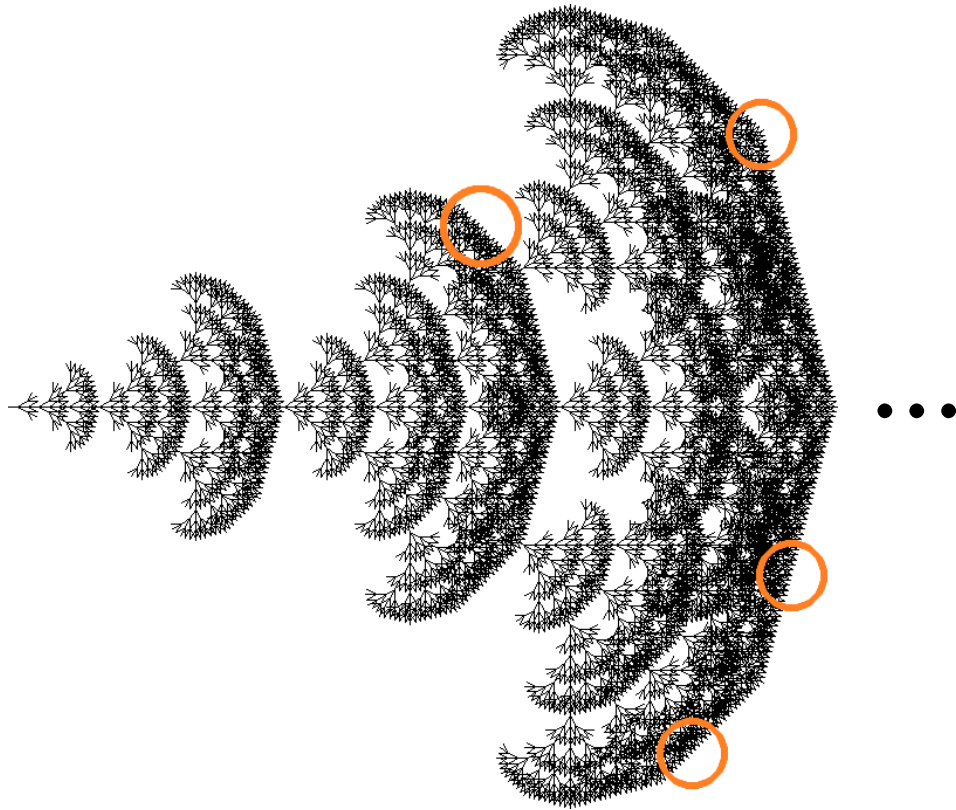
$$\begin{array}{c} \text{(ax)} \frac{t \in \Gamma}{\Gamma \vdash t} \\ \text{(ext)} \frac{}{\Gamma \vdash rtr} \\ \text{(ext)} \frac{}{\Gamma \vdash artra} \\ \text{(ext)} \frac{}{\Gamma \vdash tartrat} \\ \text{(ext)} \frac{}{\Gamma \vdash etartrate} \\ \text{(ext)} \frac{}{\Gamma \vdash detartrated} \\ \text{(concat)} \frac{}{\Gamma \vdash \text{radardetartratedradar}} \end{array} \quad \begin{array}{c} \text{(ax)} \frac{d \in \Gamma}{\Gamma \vdash d} \\ \text{(ext)} \frac{}{\Gamma \vdash ada} \\ \text{(ext)} \frac{}{\Gamma \vdash radar} \end{array}$$



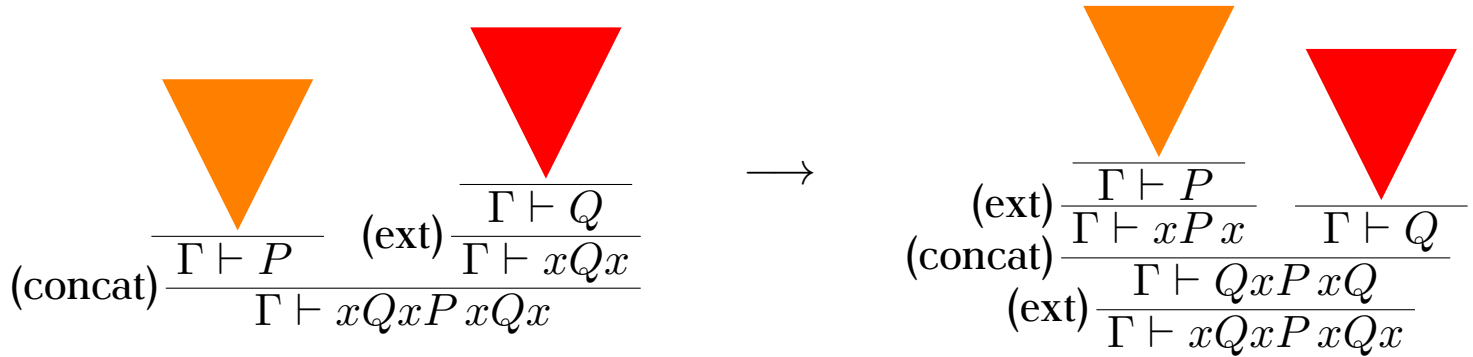
Tree of proofs



Tree of proofs



Proof reduction



Proof reduction: example

$$\begin{array}{c} \text{(ax)} \frac{t \in \Gamma}{\Gamma \vdash t} \\ \text{(ext)} \frac{\Gamma \vdash t}{\Gamma \vdash rtr} \\ \text{(ext)} \frac{\Gamma \vdash rtr}{\Gamma \vdash artra} \\ \text{(ext)} \frac{\Gamma \vdash artra}{\Gamma \vdash tartrat} \\ \text{(ext)} \frac{\Gamma \vdash tartrat}{\Gamma \vdash etartrate} \\ \text{(ext)} \frac{\Gamma \vdash etartrate}{\Gamma \vdash detartrated} \\ \text{(concat)} \frac{\Gamma \vdash detartrated}{\Gamma \vdash radardetartratedradar} \end{array} \quad \begin{array}{c} \text{(ax)} \frac{d \in \Gamma}{\Gamma \vdash d} \\ \text{(ext)} \frac{\Gamma \vdash d}{\Gamma \vdash ada} \\ \text{(ext)} \frac{\Gamma \vdash ada}{\Gamma \vdash radar} \end{array}$$



Proof reduction: example

$$\begin{array}{c}
 \text{(ax)} \frac{t \in \Gamma}{\Gamma \vdash t} \\
 \text{(ext)} \frac{\Gamma \vdash t}{\Gamma \vdash rtr} \\
 \text{(ext)} \frac{\Gamma \vdash rtr}{\Gamma \vdash artra} \\
 \text{(ext)} \frac{\Gamma \vdash artra}{\Gamma \vdash tartrat} \\
 \text{(ext)} \frac{\Gamma \vdash tartrat}{\Gamma \vdash etartrate} \\
 \text{(ext)} \frac{\Gamma \vdash etartrate}{\Gamma \vdash detartrated} \quad \text{(ax)} \frac{d \in \Gamma}{\Gamma \vdash d} \\
 \text{(ext)} \frac{\Gamma \vdash detartrated}{\Gamma \vdash rdetartratedr} \quad \text{(ext)} \frac{\Gamma \vdash d}{\Gamma \vdash ada} \\
 \text{(concat)} \frac{\Gamma \vdash rdetartratedr \quad \Gamma \vdash ada}{\Gamma \vdash adardetartratedrada} \\
 \text{(ext)} \frac{\Gamma \vdash adardetartratedrada}{\Gamma \vdash radardetartratedradar}
 \end{array}$$



Proof reduction: example

$$\begin{array}{c} \text{(ax)} \frac{t \in \Gamma}{\Gamma \vdash t} \\ \text{(ext)} \frac{\Gamma \vdash t}{\Gamma \vdash rtr} \\ \text{(ext)} \frac{\Gamma \vdash artra}{\Gamma \vdash tartrat} \\ \text{(ext)} \frac{\Gamma \vdash tartrat}{\Gamma \vdash etartrate} \\ \text{(ext)} \frac{\Gamma \vdash etartrate}{\Gamma \vdash detartrated} \\ \text{(ext)} \frac{\Gamma \vdash detartrated}{\Gamma \vdash rdetartratedr} \\ \text{(ext)} \frac{\Gamma \vdash rdetartratedr}{\Gamma \vdash ardetartratedra} \quad \text{(ax)} \frac{d \in \Gamma}{\Gamma \vdash d} \\ \text{(concat)} \frac{\Gamma \vdash ardetartratedra \quad \Gamma \vdash d}{\Gamma \vdash dardetartratedrad} \\ \text{(ext)} \frac{\Gamma \vdash dardetartratedrad}{\Gamma \vdash adardetartratedrada} \\ \text{(ext)} \frac{\Gamma \vdash adardetartratedrada}{\Gamma \vdash radardetartratedradar} \end{array}$$

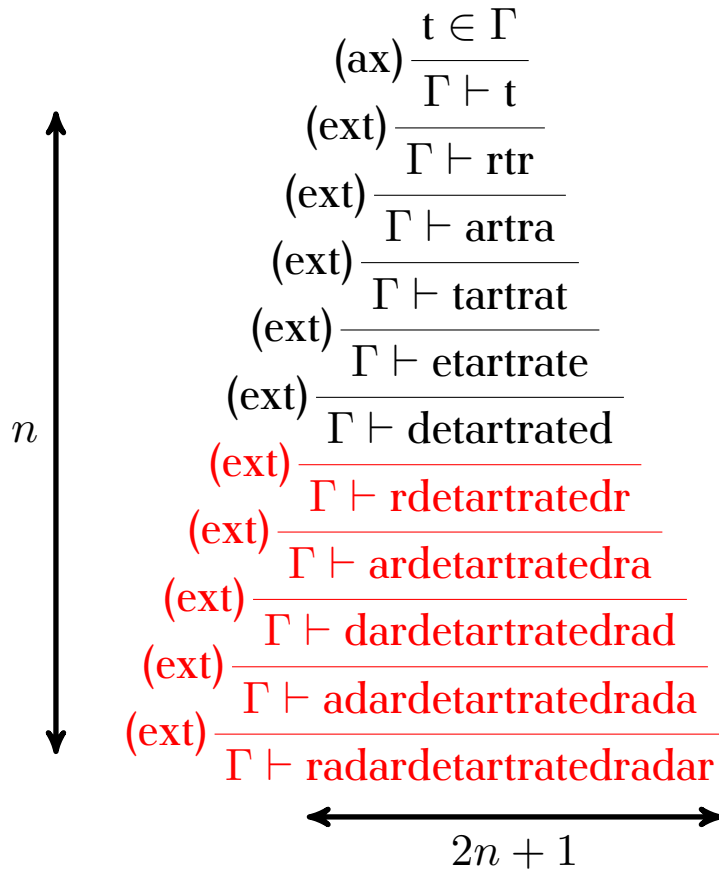


Proof reduction: example

$$\begin{array}{c} \text{(ax)} \frac{t \in \Gamma}{\Gamma \vdash t} \\ \text{(ext)} \frac{\Gamma \vdash t}{\Gamma \vdash rtr} \\ \text{(ext)} \frac{\Gamma \vdash rtr}{\Gamma \vdash artra} \\ \text{(ext)} \frac{\Gamma \vdash artra}{\Gamma \vdash tartrat} \\ \text{(ext)} \frac{\Gamma \vdash tartrat}{\Gamma \vdash etartrate} \\ \text{(ext)} \frac{\Gamma \vdash etartrate}{\Gamma \vdash detartrated} \\ \text{(ext)} \frac{\Gamma \vdash detartrated}{\Gamma \vdash rdetartratedr} \\ \text{(ext)} \frac{\Gamma \vdash rdetartratedr}{\Gamma \vdash ardetartratedra} \\ \text{(ext)} \frac{\Gamma \vdash ardetartratedra}{\Gamma \vdash dardetartratedrad} \\ \text{(ext)} \frac{\Gamma \vdash dardetartratedrad}{\Gamma \vdash adardetartratedrada} \\ \text{(ext)} \frac{\Gamma \vdash adardetartratedrada}{\Gamma \vdash radardetartratedradar} \end{array}$$



Proof reduction: example



- ☛ Proof in normal form.
- ☛ Proof always ends with an (ax) or (ext) rule.
- ☛ Can compute with proofs.

$\Gamma, \text{radar} \vdash \text{radar}$



Modus Ponens

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$




Computation with proofs of logical formulae

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$


$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$$

Computation Rule:



$$\frac{\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \quad \Gamma \vdash A}{\Gamma \vdash B}$$

→



$$\frac{\Gamma \vdash B}{\Gamma \vdash B}$$



Modulo --- formula rewriting

Proofs \longleftrightarrow Programs
Formulae \longleftrightarrow Types

- ☛ Want to reason on **proofs/programs**.
- ☛ If we can write proofs inside formulae then we should be able to **compute** inside formulae.
- ☛ Computation is a means to reduce proof effort (e.g. Four Colour Theorem, reflexive tactics).

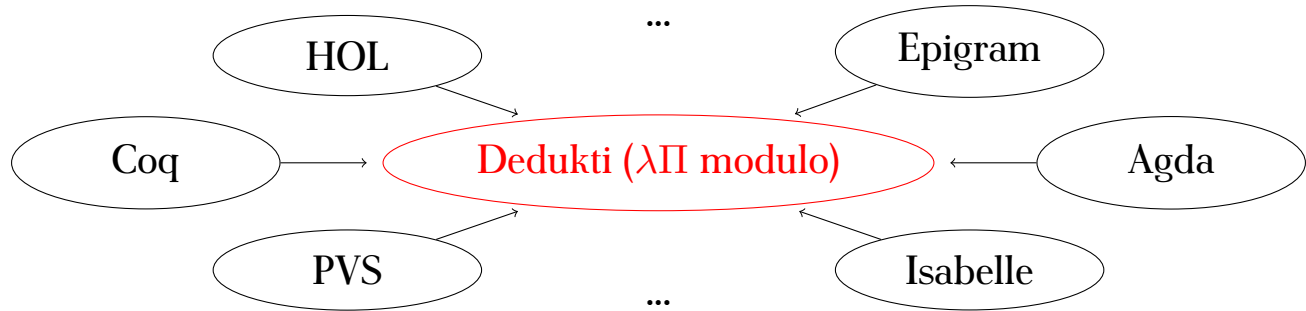


Dedukti

Dedukti ($\lambda\Pi$ modulo)



Dedukti



Thesis

*Analysis, transformation and compilation of programs
is a simple and effective method for checking proofs.*

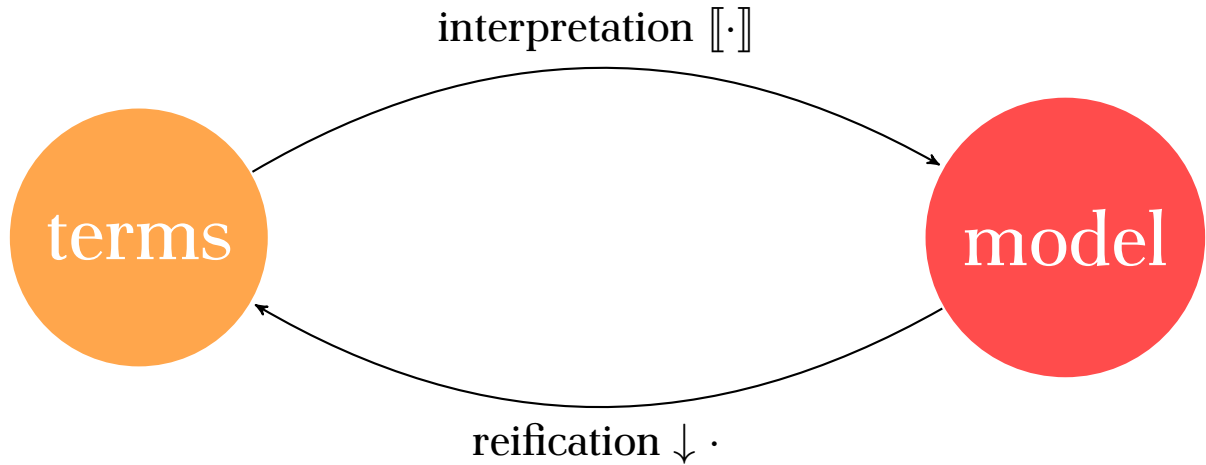


Conversion test

$$\frac{\Gamma \vdash A}{\Gamma \vdash B} \quad A \equiv_{\beta\mathcal{R}} B$$



Normalization by Evaluation



1. $\forall M. \forall N. M \equiv N \Rightarrow \llbracket M \rrbracket = \llbracket N \rrbracket$ (soundness),
2. $\forall M. \downarrow \llbracket M \rrbracket = M$ if M is in normal form (reproduction).



From program to data

$$\ulcorner x \urcorner = \mathbf{B} \ x$$

$$\ulcorner \underline{\lambda}x. M \urcorner = \mathbf{Lam} \ (\lambda x. \ulcorner M \urcorner)$$

$$\ulcorner M \ \underline{\cdot} \ N \urcorner = \mathbf{App} \ \ulcorner M \urcorner \ \ulcorner N \urcorner$$



Data evaluation

$$\lceil x \rceil = \text{B } x$$

$$\lceil \lambda x. M \rceil = \text{Lam } (\lambda x. \lceil M \rceil)$$

$$\lceil M _ N \rceil = \text{App } \lceil M \rceil \lceil N \rceil$$

$$\text{eval } (\text{B } x) = x$$

$$\text{eval } (\text{Lam } f) = \lambda x. \text{eval } (f \ x)$$

$$\text{eval } (\text{App } M \ N) = \text{app } (\text{eval } M) (\text{eval } N)$$

$$\text{app } f \ N = f \ N$$



Evaluation to a residualizing semantics

$$\lceil x \rceil = \mathbf{B} \ x$$

$$\lceil \lambda x. M \rceil = \mathbf{Lam} \ (\lambda x. \lceil M \rceil)$$

$$\lceil M _ N \rceil = \mathbf{App} \ \lceil M \rceil \ \lceil N \rceil$$

$$\mathbf{eval} \ (\mathbf{B} \ x) = x$$

$$\mathbf{eval} \ (\mathbf{Lam} \ f) = \mathbf{Lam} \ (\lambda x. \mathbf{eval} \ (f \ x))$$

$$\mathbf{eval} \ (\mathbf{App} \ M \ N) = \mathbf{app} \ (\mathbf{eval} \ M) \ (\mathbf{eval} \ N)$$

$$\mathbf{app} \ (\mathbf{Lam} \ f) \ N = f \ N$$

$$\mathbf{app} \ M \ N = \mathbf{App} \ M \ N$$



Interpretation

$$\ulcorner x \urcorner = \mathbf{B} \ x$$

$$\ulcorner \lambda x. M \urcorner = \mathbf{Lam} \ (\lambda x. \ulcorner M \urcorner)$$

$$\ulcorner M _ N \urcorner = \mathbf{App} \ \ulcorner M \urcorner \ \ulcorner N \urcorner$$

$$\mathbf{eval} \ (\mathbf{B} \ x) = x$$

$$\mathbf{eval} \ (\mathbf{Lam} \ f) = \mathbf{Lam} \ (\lambda x. \mathbf{eval} \ (f \ x))$$

$$\mathbf{eval} \ (\mathbf{App} \ M \ N) = \mathbf{app} \ (\mathbf{eval} \ M) \ (\mathbf{eval} \ N)$$

$$\mathbf{app} \ (\mathbf{Lam} \ f) \ N = f \ N$$

$$\mathbf{app} \ M \ N = \mathbf{App} \ M \ N$$

$$\llbracket M \rrbracket = \mathbf{eval} \ \ulcorner M \urcorner.$$



Partial evaluation of $\text{eval} \circ \ulcorner \cdot \urcorner$

$$\llbracket x \rrbracket = x$$

$$\llbracket \lambda x. M \rrbracket = \text{Lam } (\lambda x. \llbracket M \rrbracket)$$

$$\llbracket M _ N \rrbracket = \text{app } \llbracket M \rrbracket \llbracket N \rrbracket$$



Reification

$$\llbracket x \rrbracket = x$$

$$\llbracket \lambda x. M \rrbracket = \text{Lam } (\lambda x. \llbracket M \rrbracket)$$

$$\llbracket M _ N \rrbracket = \text{app } \llbracket M \rrbracket \llbracket N \rrbracket$$

$$\downarrow_n \mathbf{F} m = m$$

$$\downarrow_n \text{Lam } f = \underline{\lambda} n. \downarrow_{n+1} (f (\mathbf{F} n))$$

$$\downarrow_n \text{App } M N = (\downarrow_n M) _ (\downarrow_n N)$$



Rewrite Rules and extensions

$$\llbracket _ \rrbracket = _$$

$$\llbracket x \rrbracket = x$$

$$\llbracket c P_1 \dots P_n \rrbracket = \text{App } (\dots (\text{App } (\text{Con } \hat{c}) \llbracket P_1 \rrbracket) \dots) \llbracket P_n \rrbracket$$

fix $(\lambda c. \lambda x_1. \dots \lambda x_n.$

case (x_1, \dots, x_n) **of**

$$(\llbracket P_{11} \rrbracket, \dots, \llbracket P_{1n} \rrbracket) \rightarrow \llbracket M_1 \rrbracket$$

$$\vdots$$

$$(\llbracket P_{m1} \rrbracket, \dots, \llbracket P_{mn} \rrbracket) \rightarrow \llbracket M_m \rrbracket$$

default

\rightarrow

$$\text{App } (\dots (\text{App } (\text{Con } \hat{c}) x_1) \dots) x_n$$

$$\left[\begin{array}{l} c P_{11} \dots P_{1n} \longrightarrow M_1 \\ \vdots \\ c P_{m1} \dots P_{mn} \longrightarrow M_m \end{array} \right] =$$

- Untyped NbE extends naturally to residual forms and reduction rules of the Calculus of Constructions.

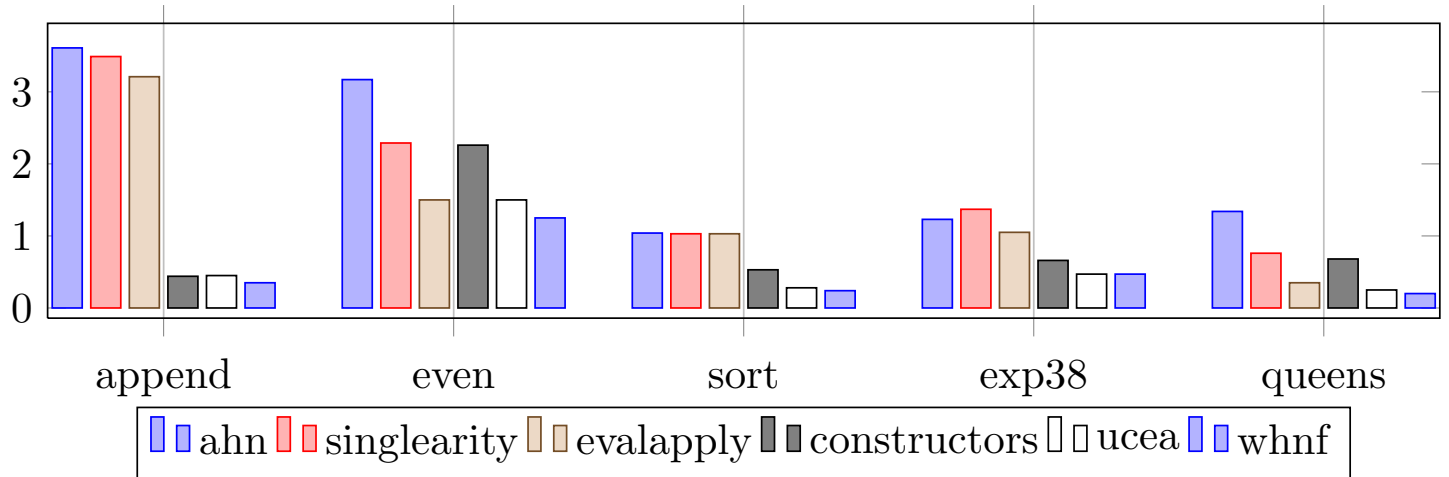


Optimizations

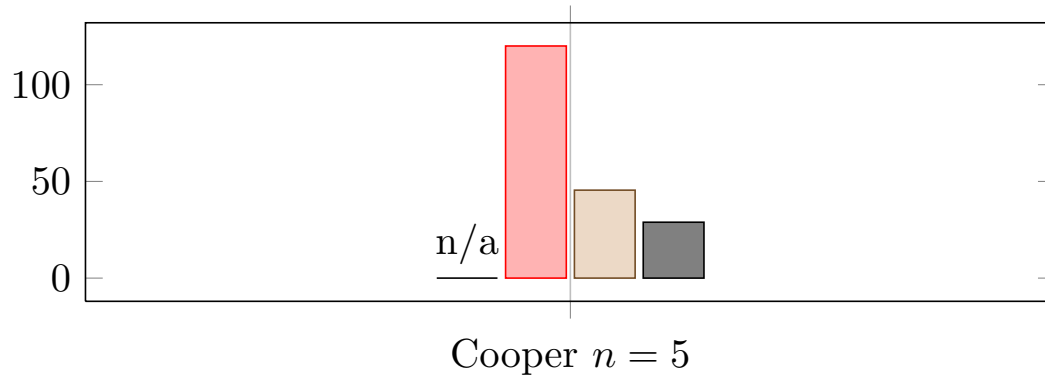
- ☛ Removal of intermediate closure allocation by standard eval/apply transformation.
- ☛ Constructors of object-level datatypes interpreted as metalevel constructors.
- ☛ Native pattern matching.



Micro benchmarks



Synthetic benchmark



Standard VM NbE NbE accu



Context-free typing



An alternative interpretation

$$\llbracket x \rrbracket = x$$

$$\llbracket \underline{\lambda}x. M \rrbracket = \text{Lam } (\lambda x. \llbracket M \rrbracket)$$

$$\llbracket M \underline{\cdot} N \rrbracket = \text{App } \llbracket M \rrbracket \llbracket N \rrbracket$$



Dependent product elimination

$$\text{(app)} \frac{\Gamma \vdash M : \Pi x : A. B \quad \Gamma \vdash N : A}{\Gamma \vdash M N : \{N / x\} B}$$



Dependent product elimination

$$\text{(app)} \frac{\Gamma \vdash M : \Pi x : A. B \quad \Gamma \vdash N : A}{\Gamma \vdash M N : \{N/x\}B}$$

$$\text{(app-ho)} \frac{\Gamma \vdash M : \Pi i A \ f \quad \Gamma \vdash N : A}{\Gamma \vdash M N : f \ N}$$

- Easy implementation of capture avoiding substitution.

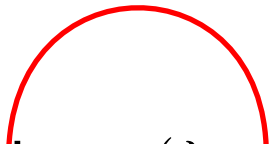


Dependent product introduction

$$\llbracket \underline{\lambda}x. M \rrbracket = \text{Lam } (\lambda x. \llbracket M \rrbracket)$$



Dependent product introduction

$$\llbracket \underline{\lambda}x. M \rrbracket = \text{Lam } \underbrace{(\lambda x. \llbracket M \rrbracket)}_f$$




Dependent product introduction

f (Var n)



Dependent product introduction

$$f \ [n : A]$$



Dependent product introduction

$$\text{(abs)} \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A. M : \Pi x : A. B}$$



Dependent product introduction

$$\text{(abs)} \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A. M : \Pi x : A. B}$$

$$\text{(abs-ho)} \frac{\vdash M : f [n : A]}{\vdash \text{Lam } A \ f : \text{Pi } A \ f}$$

☛ Drop explicit context in judgements.



Towards a LCF style proof checker for dependently typed theories

- ☛ Type decorated variable occurrences in HOL.
- ☛ Proofs checked by construction.
- ☛ Allows cheap combination of proofs.
- ☛ No context — no checking that contexts are compatible.

Example:



$$\frac{\quad}{\Gamma \vdash M : \Pi x : A. B} \quad \frac{\quad}{\Gamma' \vdash N : A'}$$



Towards a LCF style proof checker for dependently typed theories

- ☛ Type decorated variable occurrences in HOL.
- ☛ Proofs checked by construction.
- ☛ Allows cheap combination of proofs.
- ☛ No context — no checking that contexts are compatible.

Example:


$$\frac{\frac{\Gamma \vdash M : \Pi x : A. B}{\Gamma \vdash \{N/x\}B} \quad \frac{\Gamma' \vdash N : A'}{\Gamma \vdash \{N/x\}B}}{\Gamma \vdash \{N/x\}B}$$



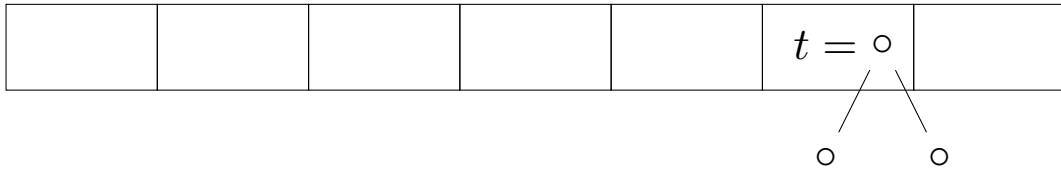
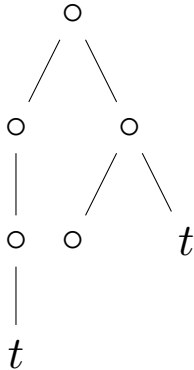
A purely functional kernel

- Proof checked by construction means no need for global registry of checked proofs.
- No state during proof checking.



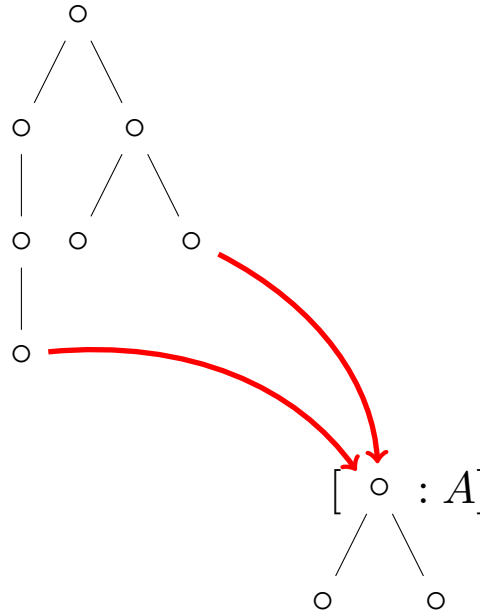
A purely functional kernel

- Proof checked by construction means no need for global registry of checked proofs.
- No state during proof checking.



A purely functional kernel

- Proof checked by construction means no need for global registry of checked proofs.
- No state during proof checking.



Managing dual interpretations



Code explosion: example

$a_1 (a_2 (a_3 (a_4 (a_5 (a_6 (a_7 a_8))))))$

A dense, repetitive pattern of vertical lines, resembling a barcode or a highly compressed representation of the nested structure above. The lines are of varying heights and are arranged in a way that suggests a complex, multi-layered structure, possibly representing the expansion of the nested parentheses in the text above.

Recuperating sharing

$$\llbracket x \rrbracket_\rho = \rho(x) \quad \text{si } x \in \text{dom}(\rho).$$

$$\llbracket s \rrbracket_\rho = \langle s, s \rangle$$

$$\llbracket \lambda x : A. M \rrbracket_\rho = \text{Let } \llbracket A \rrbracket_\rho (\bar{\lambda}y. \langle \text{Lam } \hat{y} (\bar{\lambda}x. \llbracket M \rrbracket_{\rho[x \mapsto x]}), \text{Lam } (\bar{\lambda}x. \overline{\llbracket M \rrbracket}) \rangle \rangle)$$

$$\llbracket \Pi x : A. B \rrbracket_\rho = \text{Let } \llbracket A \rrbracket_\rho (\bar{\lambda}y. \langle \text{Pi } \hat{y} (\bar{\lambda}x. \llbracket B \rrbracket_{\rho[x \mapsto x]}), \text{Pi } \check{y} (\bar{\lambda}x. \overline{\llbracket B \rrbracket}) \rangle \rangle)$$

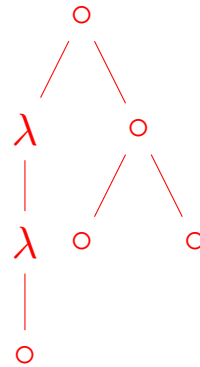
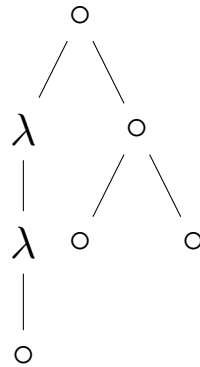
$$\llbracket M N \rrbracket_\rho = \text{Let } \llbracket N \rrbracket_\rho (\bar{\lambda}x. \text{Let } \llbracket M \rrbracket_\rho (\bar{\lambda}y. \langle \text{App } \hat{x} \hat{y}, \text{app } \check{x} \check{y} \rangle \rangle))$$



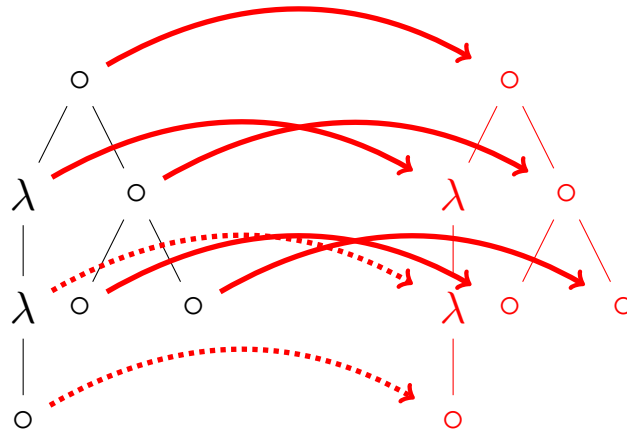
Connecting subterms to their code



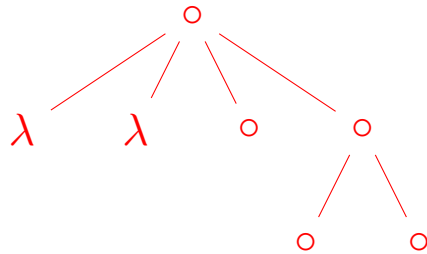
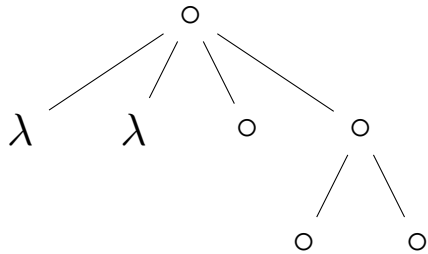
Connecting subterms to their code



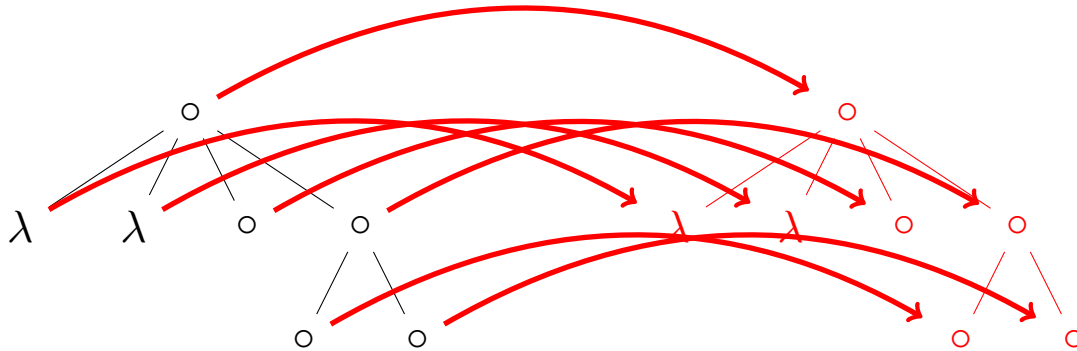
Connecting subterms to their code



Lambda-lifting



Lambda-lifting



|||||

Final words



Proof checking by program analysis,
transformation and compilation is a cheap
and effective method for checking proofs.



Future work

- More clever shortcutting of normalization.
- Development of more embeddings in the $\lambda\Pi$ -calculus modulo.
- Bootstrap of core type checker.

