Future directions

Roadmap

Cedille 1.0.0

Roadmap

Datatype notations (1.1)

$\uparrow$

Cedille 1.0.0

Roadmap

2nd order matching (1.2)

$\uparrow$

Datatype notations (1.1)

$\uparrow$

Cedille 1.0.0

Roadmap

extensionality (1.3)

$\uparrow$

2nd order matching (1.2)

$\uparrow$

Datatype notations (1.1)

$\uparrow$

Cedille 1.0.0

Roadmap

sequent calculus (2.0)

$\uparrow$

extensionality (1.3)

$\uparrow$

2nd order matching (1.2)

$\uparrow$

Datatype notations (1.1)

$\uparrow$

Cedille 1.0.0

Roadmap

sequent calculus (2.0)

$\uparrow$

$\boxed{\text{extensionality (1.3)}}$

$\uparrow$

2nd order matching (1.2)

$\uparrow$

Datatype notations (1.1)

$\uparrow$

Cedille 1.0.0

Datatype notations (Cedille 1.1)

- High-level notation (like Coq/Agda) for
  - Declaring datatypes
  - Pattern-matching recursion
- Elaboration to pure lambda calculus!
- The theory we have supports
  - Provably monotone datatypes
  - Histomorphic recursion

Seeking extensionality

$$\forall\ x : A.\ \{f\ x \simeq g\ x\}$$

$$\Downarrow$$

$$\{f \simeq g\}$$

In extensional MLTT

$$\forall\ x : A.\ \{f\ x \simeq_B g\ x\}$$

$$\Downarrow$$

$$x : A \vdash \{f\ x \simeq_B g\ x\}$$

$$\Downarrow$$

$$\{\lambda x. f\ x \simeq_{A \to B} \lambda x. g\ x\}$$

$$\Downarrow$$

$$\{f \simeq_{A \to B} g\}$$

In extensional MLTT

$$\forall\ x : A.\ \{f\ x \simeq_{B} g\ x\}$$
$$\Downarrow$$
$$x : A \vdash \{f\ x \simeq_{B} g\ x\}$$
$$\Downarrow$$
$$\{\lambda x. f\ x \simeq_{A \to B} \lambda x. g\ x\}$$
$$\Downarrow$$
$$\{f \simeq_{A \to B} g\}$$

$\boxed{\text{Seems to require typed equality!}}$

Extending Cedille's equality

- Currently:

$$\{t \simeq t'\}$$

$$[\![\{t \simeq t'\}]\!] \quad = \quad \{\hat{t} \mid [\![t]\!] =_{\beta\eta} [\![t']\!]\}$$

- Proposed:

$$\{t \simeq t' \mathbin{@} T\}$$

$$\begin{array}{lcl} [\![\{t \simeq t' \mathbin{@} T\}]\!] & = & Eq_T\ t\ t' \\ Eq_{A \to B}\ t\ t' & \Leftrightarrow & \forall\ t_1 \in [\![A]\!].\ Eq_B\ (t\ t_1)\ (t'\ t_1) \\ \ldots & & \end{array}$$

$\boxed{\text{Terms equal at } T \text{ iff indistinguishable by } T\text{-contexts}}$

This is not ETT

In ETT:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash t' : A}{\Gamma \vdash t =_A t' : \star}$$

In proposed extension:

$$\frac{\Gamma \vdash T : \star \qquad FV(t\ t') \subseteq dom(\Gamma)}{\Gamma \vdash \{t \simeq t' \ @\ T\} : \star}$$

(Strange) Example

$$True \;=\; \forall\, X : \star.\, X \to X$$

The following are equal at type $True \to True$:

$$\lambda\, s.\, \lambda\, z.\, s\; (s\; (z\; \lambda\, q.\, q))$$

$$\lambda\, s.\, \lambda\, z.\, s\; (z\; \lambda\, q.\, q)$$

- ▷ Neither term has type $True \to True$
- ▷ Applied to $\lambda\, x.\, x$, they are β-equal

Why?

- ▷ General considerations, and

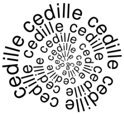
- ▷ Certain examples, including higher-order abstract syntax

$$\mathit{LamTrm} \;:\; \star \;=\; \forall\, X : \star.((X \to X) \to X) \to (X \to X \to X) \to X$$

Sequent calculus?

- Good formalism for duality
- Under CH, supports control
- Challenge: retain canonicity
  - BiInt adds dual to subtraction, loses disjunction property
  - With T. Cantor: logic **2Int**^{x}
- Motivation: coinduction dual to induction

Recap

ç?	Motivation and background for Cedille
$\vdash$ *CeDiLlE*	Syntax and semantics
`cedille`	Tooling: emacs frontend $\leftrightarrow$ backend
$\rightsquigarrow$ `cedille`$_{\texttt{core}}$	Elaboration to Cedille Core
`c d ll`	Spine-local type inference
	Future directions

Thanks

Rest of Cedille dev:

Anthony Cantor, Larry Diehl, Andrew Marmaduke

Denis Firsov

U. Iowa Computational Logic Center (Tinelli, Chowdhury, Reynolds)

Funders: NSF, AFOSR(MURI)