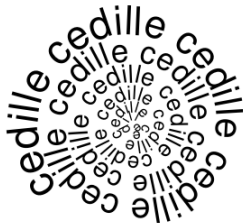


Lab

`github.com/cedille/icfp18-tutorial/tree/
master/lab`



Future directions

Roadmap

Cedille 1.0.0

Roadmap

Datatype notations (1.1)



Cedille 1.0.0

Roadmap

2nd order matching (1.2)

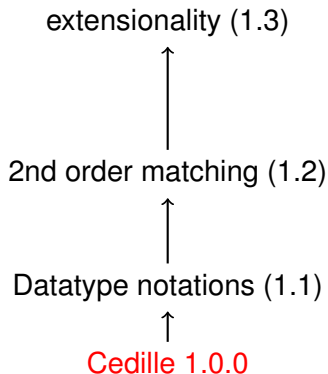


Datatype notations (1.1)

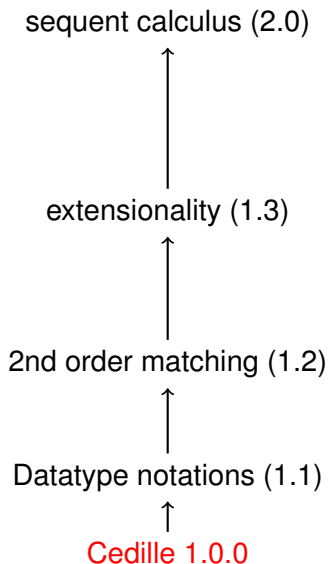


Cedille 1.0.0

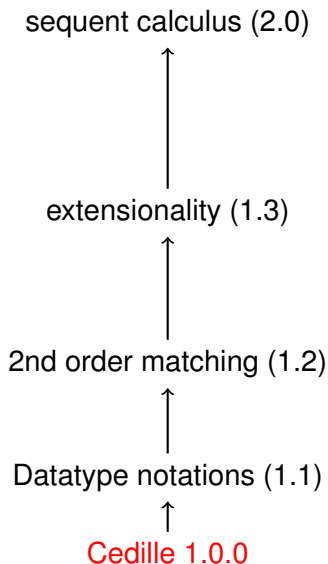
Roadmap



Roadmap



Roadmap



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

Cedilleum \subseteq Cedille, for Victor Maia of Ethereum Foundation

Seeking extensionality

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



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

In extensional MLTT

$$\begin{aligned} & \forall x : A. \{f\ x \simeq_B g\ x\} \\ & \Downarrow \\ & x : A \vdash \{f\ x \simeq_B g\ x\} \\ & \Downarrow \\ & \{\lambda x. f\ x \simeq_{A \rightarrow B} \lambda x. g\ x\} \\ & \Downarrow \\ & \{f \simeq_{A \rightarrow B} g\} \end{aligned}$$

In extensional MLTT

$$\begin{aligned} & \forall x : A. \{f\ x \simeq_B g\ x\} \\ & \Downarrow \\ & x : A \vdash \{f\ x \simeq_B g\ x\} \\ & \Downarrow \\ & \{\lambda x. f\ x \simeq_{A \rightarrow B} \lambda x. g\ x\} \\ & \Downarrow \\ & \{f \simeq_{A \rightarrow B} g\} \end{aligned}$$

Seems to require typed equality!

Extending Cedille's equality

- ▷ Currently:

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

$$\llbracket \{t \simeq t'\} \rrbracket = \{\hat{t} \mid \llbracket t \rrbracket =_{\beta\eta} \llbracket t' \rrbracket\}$$

- ▷ Proposed:

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

$$\begin{aligned} \llbracket \{t \simeq t' @ T\} \rrbracket &= Eq_T t t' \\ Eq_{A \rightarrow B} t t' &\Leftrightarrow \forall t_1 \in \llbracket A \rrbracket. Eq_B (t t_1) (t' t_1) \\ &\dots \end{aligned}$$

Terms equal at T iff indistinguishable by T -contexts

This is not ETT

In ETT:

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

In proposed extension:

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

(Strange) Example

$$True = \forall X : *. X \rightarrow X$$

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

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

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

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

Why?

- ▷ General considerations, and
- ▷ Certain examples, including higher-order abstract syntax

$$\text{LamTrm} : \star = \forall X : \star. ((X \rightarrow X) \rightarrow X) \rightarrow (X \rightarrow X \rightarrow X) \rightarrow X$$

Sequent calculus?

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

Recap



Motivation and background for Cedille

\vdash *CeDiLLe*

Syntax and semantics

`cedille`

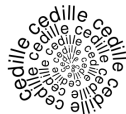
Tooling: emacs frontend \leftrightarrow backend

\leadsto `cedillecore`

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)

