What is Cedille$_{\text{core}}$?

- Independent implementation of CDLE (fact check?)
- Under 1000 lines of code, in Haskell
- Austere sublanguage of Cedille
- Cedille sanity check

Cedille vs. Cedille$_{\text{core}}$

$(\lambda\ x.\ t)$

$(\lambda\ x : T.\ t)$

$[\ t_1,\ t_2\ ]$

δ

Type Checking

$(\rho\ e\ @\ x.\ T_x\ -\ t)$

ϕ β

$[\ t_1,\ t_2\ @\ x.\ T_x\ ]$

Scope Shadowing

ε

$(\beta<u_T>\{u_t\})$

$(\beta\{u_t\})$

Type Synthesis

ς

χ

θ $(\rho<3\ 4\ 6>\ e\ -\ t)$

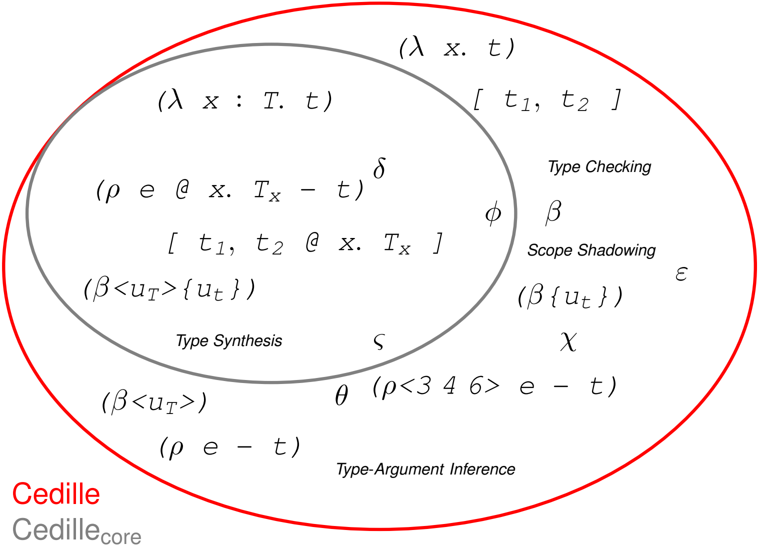
$(\beta<u_T>)$

$(\rho\ e\ -\ t)$

Type-Argument Inference

Cedille

Cedille vs. Cedille$_{\text{core}}$



Elaboration

- Classify λ-terms

  $\lambda\ x.\ t \rightsquigarrow \lambda\ x : T.\ t$

Elaboration

- Classify λ-terms

  $\lambda\ \mathtt{x}.\ \mathtt{t} \rightsquigarrow \lambda\ \mathtt{x} : \mathtt{T}.\ \mathtt{t}$

- Insert infered type arguments

  $\mathtt{id}\ \mathtt{zero} \rightsquigarrow \mathtt{id} \cdot \mathtt{Nat}\ \mathtt{zero}$

Elaboration

- Classify λ-terms

  $\lambda\ x.\ t \leadsto \lambda\ x : T.\ t$

- Insert infered type arguments

  $id\ zero \leadsto id \cdot Nat\ zero$

- Specify where rewrites occur in ρ-terms

  $? \leadsto ?$

Elaboration

- Classify λ-terms

  $\lambda\ x.\ t \rightsquigarrow \lambda\ x : T.\ t$

- Insert infered type arguments

  $id\ zero \rightsquigarrow id \cdot Nat\ zero$

- Specify where rewrites occur in ρ-terms

  $? \rightsquigarrow ?$

- Provide dependent type of second projection in ι-pairs

  $[\ zero',\ zeroi\ ] \rightsquigarrow [\ zero',\ zeroi\ @\ x.\ T_x\ ]$

Elaboration

- Classify λ-terms

  $\lambda\ x.\ t \leadsto \lambda\ x : T.\ t$

- Insert infered type arguments

  $id\ zero \leadsto id \cdot Nat\ zero$

- Specify where rewrites occur in ρ-terms

  $?\ \leadsto\ ?$

- Provide dependent type of second projection in ι-pairs

  $[\ zero',\ zeroi\ ] \leadsto [\ zero',\ zeroi\ @\ x.\ T_x\ ]$

- Ensure β-terms supply both a term t for proving $\{\ t \simeq t\ \}$ and one for erasure

  $\beta \leadsto \beta\texttt{<}t\texttt{>}\{\lambda\ x.\ x\}$

Elaboration

- Classify λ-terms

  $\lambda\ x.\ t \rightsquigarrow \lambda\ x : T.\ t$
- Insert infered type arguments

  $id\ zero \rightsquigarrow id \cdot Nat\ zero$
- Specify where rewrites occur in ρ-terms

  $? \rightsquigarrow ?$
- Provide dependent type of second projection in ι-pairs

  $[\ zero',\ zeroi\ ] \rightsquigarrow [\ zero',\ zeroi\ @\ x.\ T_x\ ]$
- Ensure β-terms supply both a term t for proving $\{\ t \simeq t\ \}$ and one for erasure

  $\beta \rightsquigarrow \beta<t>\{\lambda\ x.\ x\}$
- Convert δ-contradictions to Church-encoded $\{\ tt \simeq ff\ \}$

  $tt = \lambda\ x.\ \lambda\ y.\ x,\ ff = \lambda\ x.\ \lambda\ y.\ y,\ zero = \lambda\ f.\ \lambda\ x.\ x,\ suc = \lambda\ n.\ \lambda\ f.\ \lambda\ x.\ s\ (n\ f\ x)$

  $\{suc\ n \simeq zero\} \rightsquigarrow \{suc\ n\ (\lambda\ \_.\ tt)\ ff \simeq zero\ (\lambda\ \_.\ tt)\ ff\}$

Elaboration

- Classify λ-terms

  $\lambda\ x.\ t \leadsto \lambda\ x : T.\ t$

- Insert infered type arguments

  $id\ zero \leadsto id \cdot Nat\ zero$

- Specify where rewrites occur in ρ-terms

  $? \leadsto ?$

- Provide dependent type of second projection in ι-pairs

  $[\ zero',\ zeroi\ ] \leadsto [\ zero',\ zeroi\ @\ x.\ T_x\ ]$

- Ensure β-terms supply both a term t for proving $\{\ t \simeq t\ \}$ and one for erasure

  $\beta \leadsto \beta<t>\{\lambda\ x.\ x\}$

- Convert δ-contradictions to Church-encoded $\{\ tt \simeq ff\ \}$

  $tt = \lambda\ x.\ \lambda\ y.\ x,\ ff = \lambda\ x.\ \lambda\ y.\ y,\ zero = \lambda\ f.\ \lambda\ x.\ x,\ suc = \lambda\ n.\ \lambda\ f.\ \lambda\ x.\ s\ (n\ f\ x)$

  $\{suc\ n \simeq zero\} \leadsto \{suc\ n\ (\lambda\ \_.\ tt)\ ff \simeq zero\ (\lambda\ \_.\ tt)\ ff\}$

- Expand module parameters and import arguments