an Intensional Dependent Type Theory with Type-in-Type and Recursion

March 4, 2021

1 Examples

1.1 pretending $\star =_\star \bot$

spoofing an equality

$(\lambda pr : (\star =_\star \bot).pr\ (\lambda x.x)\ \bot \quad : \neg\star =_\star \bot)\, refl_{\star:\star}$

elaborates to

$(\lambda pr : (\star =_\star \bot).pr\ (\lambda x.x)\ \bot \quad : \neg\star =_\star \bot)\, (refl_{\star:\star} :: (\star =_\star \star) =_l (\star =_\star \bot))$

$refl_{\star:\star} :: (\star =_\star \star) =_l (\star =_\star \bot)\ (\lambda x.x)\ \bot \quad :\bot$

$(\lambda C : (\star \to \star).\lambda x : C\ \star.x :: (\Pi C : (\star \to \star).C\star \to C\star) =_l (\Pi C : (\star \to \star).C\star \to C\ \bot))\ (\lambda x.x)\ \bot$ $:\bot$

$(\lambda x : \star.x :: (\star \to \star) =_{l,bod} (\star \to \bot))\ \bot \quad :\bot$

$(\bot :: \star =_{l,bod.bod} \bot) \quad :\bot$

note that the program has not yet "gotten stuck". to exercise this error, $\bot$ must be eliminated, this can be done by tying to summon another type by applying it to $\bot$

$((\bot :: \star =_{l,bod.bod} \bot) \quad :\bot)\ \star$

$((\Pi x : \star.x) :: \star =_{l,bod.bod} (\Pi x : \star.x))\ \star$

the computation is stuck, and the original application can be blamed on account that the "proof" has a discoverable type error at the point of application l

$\lambda \Pi C : (\star \to \star).C\star \to \underline{C\star} \neq \lambda \Pi C : (\star \to \star).C\star \to \underline{C\ \bot}$

when

$C \coloneqq \lambda x.x$

$C\ \bot = \bot \neq \star = C\star$

1.2 pretending $true_c =_{\mathbb{B}_c} false_c$

spoofing an equality, evaluating $\neg true_c =_{\mathbb{B}_c} false_c$ with an incorrect proof.

$(\lambda pr : (\Pi C : (\mathbb{B}_c \to \star).C\, true_c \to C\, false_c).pr\ (\lambda b : \mathbb{B}_c.b\ \star\ \star\ \bot)\ \bot \quad : \neg true_c =_{\mathbb{B}_c} false_c)\, refl_{true_c:\mathbb{B}_c}$

is elaborated to

$(\lambda pr : (\Pi C : (\mathbb{B}_c \to \star).C\, true_c \to C\, false_c).pr\ (\lambda b : \mathbb{B}_c.b\ \star\ \star\ \bot)\ \bot \quad : \neg true_c =_{\mathbb{B}_c} false_c)\, (refl_{true_c:\mathbb{B}_c} ::$

$(refl_{true_c:\mathbb{B}_c} :: true_c =_{\mathbb{B}_c} true_c =_l true_c =_{\mathbb{B}_c} false_c)\ (\lambda b : \mathbb{B}_c.b\ \star\ \star\ \bot)\ \bot$

$((\lambda C : (\mathbb{B}_c \to \star) . \lambda x : C\, true_c . x) :: \Pi C : (\mathbb{B}_c \to \star) . C\, true_c \to C\, true_c =_l \Pi C : (\mathbb{B}_c \to \star) . C\, true_c \to C\, false$
$((. \lambda x : \star . x) :: \star \to \star =_{l,bod} \star \to \bot)\ \bot$
$(\bot :: \star =_{l,bod.bod} \bot)$

As in the above the example has not yet "gotten stuck". As above, applying $\star$ will discover the error, which would result in an error like

$\lambda \Pi C : (\mathbb{B}_c \to \star) . C\, true_c \to \underline{C\, true_c} \neq \lambda \Pi C : (\mathbb{B}_c \to \star) . C\, false_c \to \underline{C\, false_c}$

when

$C \coloneqq \lambda x . x$

$C\, true_c = \bot \neq \star = C\, false_c$

2 Data problems

note that the only way to induce an error is by feeding the functions more functions untill they get stuck. This inderection means that we cannot say $\underline{C \star} \neq \underline{C\ \bot}$ since $\star \neq \bot$ without giving a value to C. this means that we can't observe tem level differences directly.

The solution is likely to add dependent data where the motive can observe type level issues via term computatoin and case elimination observes data constructor discrepencies.

consider some data with

$T : \mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \star$

and some data constructor

$D : (x : \mathbb{N}) \to (y : \mathbb{N}) \to (z : \mathbb{N}) \to T\, x\, y\, z$

and an elimination like

$case\, D\, 0\, 1\, 0\ ::\ T\, 0\, 0\, 0\ <\!- : T\, x\, y\, z . case\, x + y + z\, of\, 0 . \mathbb{N} \to \mathbb{N} \mid\ - . \star >\ of \ldots$

3 Proof summery

elaboration produces a well typed term

type soundness goes through as before but, all head constructors match or blame is available in an empty ctx

4 TODO

- syntax and rules
- Example of why function types alone are underwhelming
  - pair,singleton
  - walk through of various examples
- theorem statements
  - substitution!

- proofs
  - top level summery!!
  - consistency is a little too restricted
  - clean up, the first type cast is unneeded and dropping it might make induction easier?
  - consider separating annotation evaluation and term evaluation
- more specifics about var binding
  - presented in an almost capture avoiding way
  - more explicit about the connection to heterogeneous substitution
- exposition
- archive
- paper target

5 Scratch

5.1 pretending $\lambda x.x =_{\mathbb{B}_c \to \mathbb{B}_c} \lambda x.true_c$

a difference can be observed via
$(\lambda pr : (\Pi C : ((\mathbb{B}_c \to \mathbb{B}_c) \to \star).C\ (\lambda x.x) \to C\ (\lambda x.true_c)).pr\ (\lambda f : \mathbb{B}_c \to \mathbb{B}_c.f \star \star \bot)\ \bot \quad : \neg \lambda x.x =_{\mathbb{B}_c \to \mathbb{B}}$
...
$S_{a:A} \coloneqq a$
$S_{a:A} \coloneqq \Pi P : A \to \star.P\, a \to \star$
..
$\neg\star =_\star (\star \to \star)$ is provable?
$\lambda pr : (\Pi C : (\star \to \star).C\, Unit \to C\ \bot).pr\ (\lambda x.x)\ \bot \quad : \neg\star =_\star \bot$
.
evaluating $\neg true_c =_{\mathbb{B}_c} false_c$ with an incorrect proof.
$(\lambda pr : (\Pi C : (\mathbb{B}_c \to \star).C\, true_c \to C\, false_c).pr\ (\lambda b : \mathbb{B}_c.b \star Unit\ \bot)\ tt \quad : \neg true_c =_{\mathbb{B}_c} false_c)\, refl_{true_c:\mathbb{B}}$
is elaborated to
$(\lambda pr : (\Pi C : (\mathbb{B}_c \to \star).C\, true_c \to C\, false_c).pr\ (\lambda b : \mathbb{B}_c.b \star Unit\ \bot)\ tt \quad : \neg true_c =_{\mathbb{B}_c} false_c)\, (refl_{true_c:}$
$\rightsquigarrow ((refl_{true_c:\mathbb{B}_c} :: true_c =_{\mathbb{B}_c} true_c =_l true_c =_{\mathbb{B}_c} false_c)\ (\lambda b : \mathbb{B}_c.b \star Unit\ \bot)\ tt \quad :\bot)$
de-sugars to
$(((\lambda C : (\mathbb{B}_c \to \star).\lambda x : C\, true_c.x) :: (\Pi C : (\mathbb{B}_c \to \star).C\, true_c \to C\, true_c) =_l (\Pi C : (\mathbb{B}_c \to \star).C\, true_c \to C\, fa$
$\rightsquigarrow (((\lambda x : (true_c \star Unit\ \bot).x) :: (true_c \star Unit\ \bot \to true_c \star Unit\ \bot) =_{l,bod} ((true_c \star Unit\ \bot) \to false_c \star$
def eq to
$(((\lambda x : Unit.x) :: (Unit \to Unit) =_{l,bod} (Unit \to \bot))\ tt \quad :\bot)$
$\rightsquigarrow (tt :: Unit =_{l,bod.bod} \bot \quad :\bot)$
note that the program has not yet "gotten stuck". to exercise this error, $\bot$ must be eliminated, this can be done by tying to summon another type by applying it to $\bot$

$(tt :: Unit =_{l,bod.bod} \bot \quad :\bot)\ \bot \quad :\bot$

bad attempt

$(tt :: Unit =_{l,bod.bod} \bot \quad :\bot)\ \bot \quad :\bot$

de-sugars to

$((\lambda A : \star.\lambda a : A.a) :: \Pi A : \star.A \to A =_{l,bod.bod} \Pi x : \star.x \quad :\bot)\ \bot \quad :\bot$

$\rightsquigarrow (.\lambda a.a) ::\bot\to\bot=_{l,bod.bod.bod}\bot \quad :\bot$

but still

$((.\lambda a.a) ::\bot\to\bot=_{l,bod.bod.bod}\bot \quad :\bot)\star$

$((.\lambda a.a) ::\bot\to\bot=_{l,bod.bod.bod} \Pi x : \star.x \quad :\bot)\star$

$(\star :: \star =_{l,bod.bod.bod.aty}\bot=_{l,bod.bod.bod.bod} \star)$

bad attempt

$(tt :: Unit =_{l,bod.bod} \bot \quad :\bot)\star \to \star \quad : \star \to \star$

de-sugars to

$((\lambda A : \star.\lambda a : A.a) :: \Pi A : \star.A \to A =_{l,bod.bod} \Pi x : \star.x \quad :\bot)\star \to \star \quad : \star \to \star$

$\rightsquigarrow (.\lambda a.a) :: (\star \to \star) \to (\star \to \star) =_{l,bod.bod.bod} \star \to \star \quad : \star \to \star$

not yet "gotten stuck"

$\rightsquigarrow (.\lambda a.a) :: (\star \to \star) \to (\star \to \star) =_{l,bod.bod.bod} \star \to \star \quad : \star \to \star$

bad attempt

de-sugars to

$((\lambda A : \star.\lambda a : A.a) :: \Pi A : \star.A \to A =_{l,bod.bod} \Pi x : \star.x \quad :\bot)\,\mathbb{B}_c \quad : \mathbb{B}_c$

$\rightsquigarrow (.\lambda a : \mathbb{B}_c.a) :: \mathbb{B}_c \to \mathbb{B}_c =_{l,bod.bod.bod} \mathbb{B}_c \quad : \mathbb{B}_c$

attempt

$(tt :: Unit =_{l,bod.bod} \bot \quad :\bot)\,\mathbb{B}_c \quad : \mathbb{B}_c$

de-sugars to

$((\lambda A : \star.\lambda a : A.a) :: \Pi A : \star.A \to A =_{l,bod.bod} \Pi x : \star.x \quad :\bot)\,\mathbb{B}_c \quad : \mathbb{B}_c$

$\rightsquigarrow (.\lambda a : \mathbb{B}_c.a) :: \mathbb{B}_c \to \mathbb{B}_c =_{l,bod.bod.bod} \mathbb{B}_c \quad : \mathbb{B}_c$

.

.

.

.

.

.