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

February 23, 2021

1 type soundness or blame

1.1 well formed

1.2 Weakening

1.3 Substitution

1.4 Preservation

1.5 Canonical forms

1.6 Progress

$\Diamond \vdash c : A$ implies that c is a value, there exists c' such that $c \rightsquigarrow c'$, or a static location can be blamed. and $\Diamond \vdash e : \overline{\star}$ implies that e is a value, there exists e' such that $e \rightsquigarrow e'$, or a static location can be blamed

By mutual induction on the typing derivations with the help of the canonical forms lemma

Explicitly:

cast typing

- $eq - ty - 1$ by **induction**
- $eq - ty - 2$ by **induction**

cast term typing

- c is typed by type-in-type. c is $\star$, a value
- c is typed by $\Pi - ty$. a is a value
- c is typed by the conversion rule, then by **induction**
- c is typed by the *apparent* rule, then c is $a_h :: e$ by each head typing
  - a_h cannot be typed by the variable rule in the empty context

- a_h is typed by type-in-type. a is $\star$, a value
- a_h is typed by $\Pi - ty$. a is a value
- a_h is typed by $\Pi - \mathsf{fun} - ty$. a is a value
- a_h is typed by $\Pi - app - ty$. Then a_h is $b\,a$, and there are derivations of $\Diamond \vdash b : \Pi x : A.B$, and $\Diamond \vdash a : A$ for some A and B. By **induction** a is a value, there exists a' such that $a \leadsto a'$, or blame and b is a value or there exists b' such that $b \leadsto b'$ or blame. (TODO jumping from one syntactic form to another)
  - if b is a value and a is a value, then b is $b_h :: v_{eq}$.
    - If all $A \in v_{eq}$ are in the form $\Pi x : A.B$ then $v_{eq}\, Elim_\Pi$ and $v_{eq} \downarrow$ is $\Pi x : A.B$ so b_h is $(\mathsf{fun}\, f.\, x.b')$ and the step is $((\mathsf{fun}\, f.\, x.b) :: v_{eq}\, v) :: v'_{eq} \leadsto (b :: e_B)\,[f := (\mathsf{fun}\, f.\, x.b) :: v_{eq}, x := v] :: v'_{eq}$
    - otherwise, $v_{eq} \uparrow$ is $\Pi x : A.B$ but there is some $[\star =_{l,o} \Pi x : A.B] \in v$ and l, o can be blamed
  - if b or a can construct blame then $b\,a$ can use that blame
  - if b is a value and $a \leadsto a'$ then $b\,a \leadsto b\,a'$
  - if $b \leadsto b'$ then $b\,a \leadsto b'\,a$

1.7 Type Soundness

For any well typed term in an empty context, no sequence of small step reductions will cause result in a computation to "get stuck" without blame. Either a final value will be reached, further reductions can be taken, or blame is omitted. This follows by iterating the progress and preservation lemmas.

2 elaboration embeds typing

1. $\vdash e : M$, $elab\,(M, *) = M'$, and $elab\,(e, M') = e'$ then $\vdash_c e' : M'$.

3 computation resulting in blame cannot be typed in the surface lang

1. $\vdash_c e' : M'$ and $e' \downarrow blame$ then there is no $\vdash e : M$ such that $elab\,(M, *) = M'$, $elab\,(e, M') = e'$

4 computation in the cast lang respects computation in the surface lang

1. $\vdash_c e' : *$ and $elab\,(e, *) = e'$ then
   (a) if $e' \downarrow *$ then $e \downarrow *$

(b) if $e' \downarrow (x : M') \to N'$ then $e \downarrow (x : M) \to N$

(c) if $e' \downarrow TCon\,\overline{M'}$ then $e \downarrow TCon\,\overline{M}$