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

February 5, 2021

1 Examples

logical unsoundness:

$\mathsf{fun}\, f : (x.x) \,.\, x : \star.f\, x \qquad : \Pi x : \star.x$

some constructs

while logically unsound the language is extremely expressive. The following calculus of Constructions constructs are expressible,

$a_1 =_A a_2 \coloneqq \lambda A : \star.\lambda a_1 : A.\lambda a_2 : A.\Pi C : (A \to \star)\,.C\, a_1 \to C\, a_2$

$Unit \coloneqq \Pi A : \star.A \to A$

$tt \coloneqq \lambda A : \star.\lambda a : A.a$

$\bot \coloneqq \Pi x : \star.x$

$\neg A \coloneqq \Pi A : \star..A \to \bot$

church nats:

$\mathbb{N}_c \coloneqq \Pi A : \star.(A \to A) \to A \to A$

$0_c \coloneqq \lambda A : \star.\lambda s : (A \to A).\lambda z : A.z$

$1_c \coloneqq \lambda A : \star.\lambda s : (A \to A).\lambda z : A.s\, z$

$2_c \coloneqq \lambda A : \star.\lambda s : (A \to A).\lambda z : A.s\ (s\, z)$

...

since there is type in type, a kind of large elimination is possible

$\lambda n : \mathbb{N}_c.n\ \star\ (\lambda - .U)\ \bot$

thus $\neg 1_c =_{\mathbb{N}_c} 0_c$ is provable (in a non trivial way):

$\lambda pr : (\Pi C : (\mathbb{N}_c \to \star)\,.C\, 1_c \to C\, 0_c)\,.pr\ (\lambda n : \mathbb{N}_c.n\ \star\ (\lambda - .U)\ \bot)\ tt \qquad : \neg 1_c =_{\mathbb{N}_c} 0_c$

There are more examples in [1] where Cardelli has studied a similar system.

2 Properties

2.1 Contexts

2.1.1 Sub-Contexts are well formed

The following rules are admissible:

$$\frac{\Gamma, \Gamma' \vdash}{\Gamma \vdash}$$

$$\frac{\Gamma, \Gamma' \vdash M : \sigma}{\Gamma \vdash}$$

$$\frac{\Gamma, \Gamma' \vdash M \Rrightarrow M' : \sigma}{\Gamma \vdash}$$

$$\frac{\Gamma, \Gamma' \vdash M \Rrightarrow_{*} M' : \sigma}{\Gamma \vdash}$$

$$\frac{\Gamma, \Gamma' \vdash M \equiv M' : \sigma}{\Gamma \vdash}$$

by mutual induction on the derivations.

2.1.2 Context weakening

For any derivation of $\Gamma \vdash \sigma : \star$, the following rules are admissible:

$$\frac{\Gamma, \Gamma' \vdash}{\Gamma, x : \sigma, \Gamma' \vdash}$$

$$\frac{\Gamma, \Gamma' \vdash M : \tau}{\Gamma, x : \sigma, \Gamma' \vdash M : \tau}$$

$$\frac{\Gamma, \Gamma' \vdash M \Rrightarrow M' : \sigma}{\Gamma, x : \sigma, \Gamma' \vdash M \Rrightarrow M' : \sigma}$$

$$\frac{\Gamma, \Gamma' \vdash M \Rrightarrow_{*} M' : \sigma}{\Gamma, x : \sigma, \Gamma' \vdash M \Rrightarrow_{*} M' : \sigma}$$

$$\frac{\Gamma, \Gamma' \vdash M \equiv M' : \tau}{\Gamma, x : \sigma, \Gamma' \vdash M \equiv M' : \tau}$$

by mutual induction on the derivations.

2.1.3 $\Rrightarrow$ is reflexive

The following rule is admissible:

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M \Rrightarrow M : \sigma} \Rrightarrow\text{-refl}$$

by induction

2.1.4 $\equiv$ is reflexive

The following rule is admissible:

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M \equiv M : \sigma} \equiv\text{-refl}$$

by $\Rrightarrow *$-refl

2.1.5 Context substitution

For any derivation of $\Gamma \vdash N : \tau$ the following rules are admissible:

$$\frac{\Gamma, x : \tau, \Gamma' \vdash}{\Gamma, \Gamma' [x \coloneqq N] \vdash}$$

$$\frac{\Gamma, x : \tau, \Gamma' \vdash M : \sigma}{\Gamma, \Gamma' [x \coloneqq N] \vdash M [x \coloneqq N] \,:\, \sigma [x \coloneqq N]}$$

$$\frac{\Gamma, x : \tau, \Gamma' \vdash M \Rrightarrow M' : \sigma}{\Gamma, \Gamma' [x \coloneqq N] \vdash M [x \coloneqq N] \Rrightarrow M' [x \coloneqq N] : \sigma [x \coloneqq N]}$$

$$\frac{\Gamma, x : \tau, \Gamma' \vdash M \Rrightarrow_* M' : \sigma}{\Gamma, \Gamma' [x \coloneqq N] \vdash M [x \coloneqq N] \Rrightarrow_* M' [x \coloneqq N] : \sigma}$$

$$\frac{\Gamma, x : \tau, \Gamma' \vdash M \equiv M' : \sigma}{\Gamma, \Gamma' [x \coloneqq N] \vdash M [x \coloneqq N] \;\equiv\; M' [x \coloneqq N] : \sigma [x \coloneqq N]}$$

by mutual induction on the derivations. Specifically, at every usage of x from the var rule in the original derivation, replace the usage of the var rule with the derivation of $\Gamma \vdash N : \tau$ weakened to the context of $\Gamma, \Gamma' [x \coloneqq N] \vdash N : \tau$, and apply $\Rrightarrow$-refl , $\Rrightarrow_*$-refl or $\equiv$-refl when needed.

2.2 Computation

2.2.1 $\Rrightarrow$ preserves type of source

The following rules are admissible:

$$\frac{\Gamma \vdash N \Rrightarrow N' : \tau}{\Gamma \vdash N : \tau}$$

by induction

2.2.2 $\Rrightarrow$ substitution

For any derivations of $\Gamma, x : \sigma, \Gamma' \vdash M \Rrightarrow M' : \tau$, and $\Gamma \vdash N \Rrightarrow N' : \sigma$, then following rule is admissible:

$$\frac{\Gamma, x : \sigma, \Gamma' \vdash M \Rrightarrow M' : \tau \quad \Gamma \vdash N \Rrightarrow N' : \sigma}{\Gamma, \Gamma' [x \coloneqq N] \vdash M [x \coloneqq N] \Rrightarrow M' [x \coloneqq N'] : \tau [x \coloneqq N]}$$

by induction on the $\Rrightarrow$ derivations

2.2.3 $\Rrightarrow$ is confluent

if $\Gamma \vdash M \Rrightarrow N : \sigma$ and $\Gamma \vdash M \Rrightarrow N' : \sigma$ then there exists P such that
$\Gamma \vdash N \Rrightarrow P : \sigma$ and $\Gamma \vdash N' \Rrightarrow P : \sigma$
by standard techniques

2.3 $\Rrightarrow_*$

2.3.1 $\Rrightarrow_*$ is transitive

The following rule is admissible:

$$\frac{\Gamma \vdash M \Rrightarrow_* M' : \sigma \quad \Gamma \vdash M' \Rrightarrow_* M'' : \sigma}{\Gamma \vdash M \Rrightarrow_* M' : \sigma} \Rrightarrow *\text{-trans}$$

by induction

2.3.2 $\Rrightarrow$ preserves type in source

$$\frac{\Gamma \vdash N \Rrightarrow N' : \tau}{\Gamma \vdash N' : \tau}$$

By induction on the $\Rrightarrow$ derivation with the help of the substitution lemma.

- Π-$\Rrightarrow$
  - $M'\,[x \coloneqq N', f \coloneqq (\mathsf{fun}\, f : (x.\tau)\,.\,x : \sigma.M')] \,:\, \tau\,[x \coloneqq N]$ by the substitution lemma used on the inductive hypotheses
- all other cases are trivial

2.3.3 $\Rrightarrow_*$ preserves type

The following rule is admissible:

$$\frac{\Gamma \vdash M \Rrightarrow_* M' : \sigma}{\Gamma \vdash M : \sigma}$$

by induction

$$\frac{\Gamma \vdash M \Rrightarrow_* M' : \sigma}{\Gamma \vdash M' : \sigma}$$

by induction

2.3.4 $\Rrightarrow_*$ is confluent

if $\Gamma \vdash M \Rrightarrow_* N : \sigma$ and $\Gamma \vdash M \Rrightarrow_* N' : \sigma$ then there exists P such that
$\Gamma \vdash N \Rrightarrow_* P : \sigma$ and $\Gamma \vdash N' \Rrightarrow_* P : \sigma$
Follows from $\Rrightarrow *$-trans and the confluence of $\Rrightarrow$ using standard techniques

2.3.5 $\equiv$ is symmetric

The following rule is admissible:

$$\frac{\Gamma \vdash M \equiv N : \sigma}{\Gamma \vdash N \equiv M : \sigma}\ \equiv\text{-sym}$$

trivial

2.3.6 $\equiv$ is transitive

$$\frac{\Gamma \vdash M \equiv N : \sigma \qquad \Gamma \vdash N \equiv P : \sigma}{\Gamma \vdash M \equiv P : \sigma}\ \equiv\text{-trans}$$

by the confluence of $\Rrightarrow_*$

2.3.7 $\equiv$ preserves type

The following rule is admissible:

$$\frac{\Gamma \vdash M \equiv\ M' : \sigma}{\Gamma \vdash M : \sigma}$$

by the def of $\Rrightarrow_*$

2.3.8 Regularity

The following rule is admissible:

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \sigma : \star}$$

by induction with $\equiv$-preservation for the Conv case

2.3.9 $\rightsquigarrow$ implies $\Rrightarrow$

For any derivations of $\Gamma \vdash M : \sigma$, $M \rightsquigarrow M'$

$$\Gamma \vdash M \Rrightarrow M' : \sigma$$

by induction on $\rightsquigarrow$

2.3.10 $\rightsquigarrow$ implies $\Rrightarrow_*$

For any derivations of $\Gamma \vdash M : \sigma$, $M \rightsquigarrow M'$,

$$\Gamma \vdash M \Rrightarrow_* M' : \sigma$$

since $\rightsquigarrow$ implies $\Rrightarrow$

2.3.11 $\rightsquigarrow$ implies $\equiv$

For any derivations of $\Gamma \vdash M : \sigma$, $M \rightsquigarrow M'$,

$$\Gamma \vdash M \equiv M' : \sigma$$

since $\rightsquigarrow$ implies $\Rrightarrow_*$

2.3.12 $\rightsquigarrow$ preserves type

For any derivations of $\Gamma \vdash M : \sigma$, $M \rightsquigarrow M'$,

$$\Gamma \vdash M' : \sigma$$

since $\rightsquigarrow$ implies $\equiv$ and $\equiv$ preserves types

2.4 Inversion

If $\Gamma \vdash M : \sigma$

if M is	$\star$	then		$\Gamma \vdash$
if M is	$\Pi x : \sigma'.\tau$	then there exists	$\Gamma \vdash \sigma' : \star$	$\Gamma, x : \sigma' \vdash \tau : \star$
if M is	$M' \, N$	then there exists	$\Gamma \vdash M' : \Pi x : \sigma'.\tau$	$\Gamma \vdash N : \sigma'$
if M is	$(\mathsf{fun}\, f : (x.\tau') \,.\, x : \sigma'.M')$	then there exists	$\Gamma, x : \sigma' \vdash \tau' : \star$	$\Gamma, x : \sigma', f : \Pi x : \sigma'.\tau \vdash M' :$

Each case follows from inspection on the typing rules. There are only 2 possibilities, the original typing rule and the conversion rule. The conversion rule must eventually refer to the original typing rule. In any sequence of conversion rules the Definitional Equality is preserved.

2.5 Type constructors

2.5.1 Type constructors are stable

- if $\Gamma \vdash * \Rrightarrow M : \sigma$ then M is $*$
- if $\Gamma \vdash * \Rrightarrow_* M : \sigma$ then M is $*$
- if $\Gamma \vdash \Pi x : \sigma.\tau \Rrightarrow M : \sigma$ then M is $\Pi x : \sigma'.\tau'$ for some σ', τ'
- if $\Gamma \vdash \Pi x : \sigma.\tau \Rrightarrow_* M : \sigma$ then M is $\Pi x : \sigma'.\tau'$ for some σ', τ'

by induction on the respective relations

2.5.2 Type constructors definitionaly unique

There is no derivation of $\Gamma \vdash * \equiv \Pi x : \sigma.\tau$ for any Γ, σ, τ

from $\equiv$ -Def and constructor stability

2.6 Canonical forms

If $\Diamond \vdash v : \sigma$ then

- if σ is $\star$ then v is $\star$ or $\Pi x : \sigma.\tau$
- if σ is $\Pi x : \sigma'.\tau$ for some σ', τ then v is $\mathsf{fun}\, f : (x.\tau')\,.\, x : \sigma''.P'$ for some τ', σ'', P'

By induction on the typing derivation

- Conv,
  - if σ is $\star$ then eventually, it was typed with type-in-type, or Π-F. it could not have been typed by Π-I since constructors are definitionaly unique
  - if σ is $\Pi x : \sigma'.\tau$ then eventually, it was typed with Π-I. it could not have been typed by type-in-type, or Π-F since constructors are definitionaly unique
- type-in-type, $\Diamond \vdash v : \sigma$ is $\Diamond \vdash \star : \star$
- Π-F, $\Diamond \vdash v : \sigma$ is $\Diamond \vdash \Pi x : \sigma.\tau\ : \star$
- Π-I, $\Diamond \vdash v : \sigma$ is $\Diamond \vdash \mathsf{fun}\, f : (x.\tau)\,.\, x : \sigma.M\ :\ \mathit{\Pi} x : \sigma.\tau$
- no other typing rules are applicable

2.7 Progress

$\Diamond \vdash M : \sigma$ implies that M is a value or there exists N such that $M \leadsto N$.

By direct induction on the typing derivation with the help of the canonical forms lemma

Explicitly:

- M is typed by the conversion rule, then by **induction**, M is a value or there exists N such that $M \leadsto N$
- M cannot be typed by the variable rule in the empty context
- M is typed by type-in-type. M is $\star$, a value
- M is typed by Π-F. M is $\Pi x : \sigma.\tau$, a value
- M is typed by Π-I. M is $\mathsf{fun}\, f : (x.\tau)\,.\, x : \sigma.M'$, a value
- M is typed by Π-E. M is $P\,N$ then by **inversion** there exist some σ, τ for $\Diamond \vdash P : \Pi x : \sigma.\tau$ and $\Diamond \vdash N : \sigma$. By **induction** (on the P branch of the derivation) P is a value or there exists P' such that $P \leadsto P'$. By **induction** (on the N branch of the derivation) N is a value or there exists N' such that $N \leadsto N'$

- if P is a value then by **canonical forms**, P is$\mathsf{fun}\, f : (x.\tau)\,.\,x : \sigma.P'$ and
  - if N is a value then the one step reduction is $(\mathsf{fun}\, f : (x.\tau)\,.\,x : \sigma.P')\ N \rightsquigarrow P'\,[x := N, f := \mathsf{fun}\, f : (x.\tau)\,.\,x : \sigma.M]$
  - otherwise there exists N' such that $N \rightsquigarrow N'$, and the one step reduction is $(\mathsf{fun}\, f : (x.\tau)\,.\,x : \sigma.P')\ N \rightsquigarrow (\mathsf{fun}\, f : (x.\tau)\,.\,x : \sigma.P')\ N'$
- otherwise, there exists P' such that $P \rightsquigarrow P'$ and the one step reduction is $P\,N \rightsquigarrow P'\,N$

2.8 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". Either a final value will be reached or further reductions can be taken. This follows by iterating the progress and preservation lemmas.

3 differences from implementation

differences from Agda development

- in both presentations standard properties of variables binding and substitution are assumed
- In Agda the parallel reduction relation does not track the original typing judgment. This should not matter for the proof of confluence.
- no need for a direct inversion lemma
- only proved the function part of the canonical forms lemma (all that is needed for the proof)

differences from prototype

- bidirectional, type annotations are not always needed on functions
- toplevel recursion in addition to function recursion
- type annotations are not relevant for definitional equality

4 Conjectured Properties

5 Non-Properties

- decidable type checking
- normalization/logical soundness

Definitional Equality does not preserve type constructors on the nose

If $\Gamma \vdash \sigma \equiv \sigma' : \star$ then

if σ is $\Pi x : \sigma''.\tau$ for some σ'', τ then σ' is $\Pi x : \sigma'''.\tau'$ for some σ''', τ'

counter example $\vdash \Pi x : \star.\star \equiv (\lambda x : \star.x)(\Pi x : \star.\star) : \star$

this implies the additional work in the Canonical forms lemma

References

[1] Luca Cardelli. *A Polymorphic [lambda]-calculus with Type: Type.* Technical Report, DEC SRC, 130 Lytton Avenue, Palo Alto, CA 94301. May. SRC Research Report, 1986.