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

February 12, 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. Conventional data is representable:

$\mathsf{data}\, Unit.\, \mathsf{where}\ \{tt. \to Unit\}$

$\mathsf{data}\, \mathbb{B}.\, \mathsf{where}\ \{true. \to \mathbb{B} \,|\, false. \to \mathbb{B}\}$

$\mathsf{data}\, \mathbb{N}.\, \mathsf{where}\ \{z. \to \mathbb{N} \,|\, suc\, x : \mathbb{N}. \to \mathbb{N}\}$
data abstraction allows some self-reference
$0 \coloneqq z$
$1 \coloneqq suc\, z$

$$\mathsf{data}\, Vec\, A : \star, n : \mathbb{N}.\, \mathsf{where} \left\{ \begin{array}{cc} nil\, A : \star. & \to Vec\, A\, 0 \\ cons\, A : \star, n : \mathbb{N}, - : A, - : Vec\, A\, n. & \to Vec\, A\ (suc\, n) \end{array} \right\}$$

$rep_{\mathbb{B}} \coloneqq \mathsf{fun}\, f : (n.Vec\, \mathbb{B}\, n)\,.\, n : \mathbb{N}. \mathsf{Case}_{n' : \mathbb{N} \to Vec\, \mathbb{B}\, n'}\, n\, \mathsf{of}\ \{z \Rightarrow nil\, \mathbb{B} \,|\, suc\, x \Rightarrow cons\, \mathbb{B}\, x\, true\ (f\, x)\} \qquad : \Pi n : \mathbb{N}. Vec\, \mathbb{B}\, n$

$\mathsf{data}\, Id\, A : \star, a_1 : A, a_2 : A.\, \mathsf{where}\ \{refl\, A : \star, a : A. \to Id\, A\, a\, a\}$
$a_1 =_A a_2 \coloneqq Id\, A, a_1, a_2$

$subst \coloneqq \lambda A. \lambda a_1. \lambda a_2. \lambda pr. \mathsf{Case}_{- : Id\, A, a_1, a_2. \to \Pi C : (A \to \star). C\, a_1 \to C\, a_2}\, pr\ \mathsf{of}\ \{refl\, A : \star, a : A \Rightarrow \lambda C. \lambda x : C\, a. x\}$
$subst : \Pi A : \star . \Pi a_1 : A. \Pi a_2 : A. a_1 =_A a_2 \to \Pi C : (A \to \star)\,. C\, a_1 \to C\, a_2$

$\mathsf{data}\ \perp\ .\, \mathsf{where}\ \{\,\}$

$\neg A := \mathit{\Pi} A : \star..A \to \bot$

$\neg 1 =_{\mathbb{N}} 0$ is provable (in a non trivial way):
$dec := \lambda n.\mathsf{Case}_{-:\mathbb{N}.\to\star}\, pr\, \mathsf{of}\mathsf{of}\, \{z \Rightarrow \bot \mid suc - \Rightarrow Unit\} \qquad : \mathbb{N} \to \star$
$\lambda pr.subst\, \mathbb{N}\, 1\, 0\, pr\, dec\, tt \qquad : \neg 1 =_{\mathbb{N}} 0$
Several larger examples are workable in prototype implementation

2 Properties

2.1 Contexts

2.1.1 Sub-Contexts are well formed

The following rules are admissible:

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

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

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

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

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

$$\frac{\Gamma, \Gamma' \vdash \Delta : \overline{*}}{\Gamma \vdash}$$

$$\frac{\Gamma, \Gamma' \vdash \overline{M} : \Delta}{\Gamma \vdash}$$

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

by mutual induction on the derivations.

2.1.2 Context weakening

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

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

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

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

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

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

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

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

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

by mutual induction on the derivations.

2.1.3 $\Rrightarrow$ is reflexive

The following rule is admissible:

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

$$\frac{\Gamma \vdash \overline{M} : \Delta}{\Gamma \vdash \overline{M} \Rrightarrow\ \overline{M} : \Delta} \Rrightarrow\text{-refl'}$$

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:

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

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

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

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

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

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

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

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

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 appropriate reflexivity when needed.

2.2 Computation

2.2.1 $\Rrightarrow$ preserves type of source

The following rules are admissible:

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

$$\frac{\Gamma \vdash \overline{N} \Rrightarrow \overline{N'} : \Delta}{\Gamma \vdash \overline{N} : \Delta}$$

by mutual induction

2.2.2 $\Rrightarrow$ substitution

The following rule is admissible:

$$\frac{\Gamma, \Delta, \Gamma' \vdash \overline{M} \Rrightarrow \overline{M'} : \Theta \quad \Gamma \vdash \overline{N} \Rrightarrow \overline{N'} : \Delta}{\Gamma, \Gamma' \left[\Delta \coloneqq \overline{N}\right] \vdash \overline{M} \left[\Delta \coloneqq \overline{N}\right] \Rrightarrow \overline{M'} \left[\Delta \coloneqq \overline{N'}\right] : \Theta \left[\Delta \coloneqq \overline{N}\right]}$$

by induction on the $\Rrightarrow$ derivations with the corollary

Corollary, the following rule is admissible:

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

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 $\Gamma \vdash N' \Rrightarrow P : \sigma$

and $\Gamma \vdash \overline{M} \Rrightarrow \overline{N} : \Delta$ and $\Gamma \vdash \overline{M} \Rrightarrow \overline{N'} : \Delta$ then there exists $\overline{P}$ such that $\Gamma \vdash \overline{N} \Rrightarrow \overline{P} : \Delta$ and $\Gamma \vdash \overline{N'} \Rrightarrow \overline{P} : \Delta$

by mutual induction on all possible pairs of reductions (abusing notation by suppressing Γ, σ, Δ that are constant throughout)

- Π-E-$\Rrightarrow$ and Π-$\Rrightarrow$
  - M is $(\mathsf{fun}\, f : (x.\tau) \,.\, x : \sigma.B)\ A$
  - N is $(\mathsf{fun}\, f : (x.\tau) \,.\, x : \sigma.B')\ A'$, $B \Rrightarrow B'$, $A \Rrightarrow A'$
  - N' is $B'' [x \coloneqq A'', f \coloneqq (\mathsf{fun}\, f : (x.\tau) \,.\, x : \sigma.B'')]$, $B \Rrightarrow B''$, $A \Rrightarrow A''$
  - $B \Rrightarrow B_v$, $A \Rrightarrow A_v$ by I.H
  - $(\mathsf{fun}\, f : (x.\tau) \,.\, x : \sigma.B)\ A \Rrightarrow B_v [x \coloneqq A_v, f \coloneqq (\mathsf{fun}\, f : (x.\tau) \,.\, x : \sigma.B_v)]$ by repeated $\Rrightarrow$ substitution
- D-E-$\Rrightarrow$ and D-$\Rrightarrow$
  - M is $\mathsf{Case}_{x:D\,\overline{y}.\sigma}\ \left(d\,\overline{A}\right)$ of $\left\{\overline{d_i\overline{x}_i \Rightarrow B_i\,|}\right\}$
  - N is $\mathsf{Case}_{x:D\,\overline{y}.\sigma}\ \left(d\,\overline{A'}\right)$ of $\left\{\overline{d_i\overline{x}_i \Rightarrow B_i'\,|}\right\}$, $\overline{A} \Rrightarrow \overline{A'}$, $\forall i.\, B_i \Rrightarrow B_i'$
  - N' is $B'' \left[\overline{x} \coloneqq \overline{A''}\right]$, $\overline{A} \Rrightarrow \overline{A''}$, $B \Rrightarrow B''$, $d\overline{x} \Rightarrow B \in_i \left\{\overline{d_i\overline{x}_i \Rightarrow B_i\,|}\right\}$
  - $B \Rrightarrow B_v$, $\overline{A} \Rrightarrow \overline{A_v}$ by I.H
  - $\mathsf{Case}_{x:D\,\overline{y}.\sigma}\ \left(d\,\overline{A}\right)$ of $\left\{\overline{d_i\overline{x}_i \Rightarrow B_i\,|}\right\} \Rrightarrow B_v \left[\overline{x} \coloneqq \overline{A_v}\right]$ by repeated $\Rrightarrow$ substitution
- all other reductions match, and follow immediately from induction, or are symmetric to already presented cases

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

The following rules are admissible:

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

$$\frac{\Gamma \vdash \overline{N} \Rrightarrow \overline{N'} : \Delta}{\Gamma \vdash \overline{N'} : \Delta}$$

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

- Π-$\Rrightarrow$
  - $M'\left[x := N', f := (\mathsf{fun}\, f : (x.\tau)\, .\, x : \sigma.M')\right] \,:\, \tau\left[x := N\right]$ by the substitution lemma used on the inductive hypotheses
- D-$\Rrightarrow$
  - $O'\left[\overline{x} := \overline{N'}\right] \,:\, \sigma\left[x := d\overline{x}_i, \overline{y} := \overline{N}\right]$ 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 rules are admissible:

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

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

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 $\Rrightarrow$

The following rules are admissible:

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

$$\frac{\Gamma \vdash \overline{M} : \Delta \quad \overline{M} \rightsquigarrow \overline{M'}}{\Gamma \vdash \overline{M} \Rrightarrow \overline{M'}' : \Delta}$$

by induction on $\rightsquigarrow$

2.3.10 $\rightsquigarrow$ preserves type

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

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

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

2.4 Type constructors

2.4.1 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'
- if $\Gamma \vdash D\,\overline{M} \Rrightarrow M : \sigma$ then M is $D\,\overline{M'}$ for some $\overline{M'}$
- if $\Gamma \vdash D\,\overline{M} \Rrightarrow_* M : \sigma$ then M is $D\,\overline{M'}$ for some $\overline{M'}$

by induction on the respective relations

2.4.2 Type constructors definitionaly unique

There is no derivation of

- $\Gamma \vdash * \equiv \Pi x : \sigma.\tau \ : \sigma'$ for any $\Gamma, \sigma, \tau, \sigma'$
- $\Gamma \vdash * \equiv D\,\overline{M} \ : \sigma'$ for any $\Gamma, \overline{M}, \sigma'$
- $\Gamma \vdash \Pi x : \sigma.\tau \equiv D\,\overline{M} : \sigma'$ for any $\Gamma, \sigma, \tau, \overline{M}, \sigma'$
- $\Gamma \vdash D\,\overline{M} \equiv D'\,\overline{N} : \sigma'$, $D \neq D'$ for any $\Gamma, \overline{M}, \overline{N}, \sigma'$

from $\equiv$ -Def and constructor stability

2.5 Canonical forms

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

- if σ is $\star$ then v is $\star$, $\Pi x : \sigma.\tau$ or $D\,\overline{M}$
- 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'
- if σ is $D\,\overline{M}$ for some $\overline{M}$ then v is $d_k\,\overline{v}$ for some $\mathsf{data}\ D\,\Delta\ \mathsf{where}\ \left\{\overline{d_i\,\Theta_i \to D\overline{M}_i \,|}\right\} \in \Xi$ and $d_k\,\Theta_k \to D\overline{M}_k \in \overline{d_i\,\Theta_i \to D\overline{M}_i \,|}$

By induction on the typing derivation

- Conv,
  - if σ is $\star$ then eventually, it was typed with type-in-type, Π-F, D-F,or D-F$'$. It could not have been typed by Π-I or D-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 b type-in-type, Π-F, D-F, D-F', or D-I since constructors are definitionaly unique
  - if σ is $D\,\overline{M}$ then eventually, it was typed with D-I. it could not have been typed b type-in-type, Π-F, D-F, D-F', or Π-I since constructors are definitionaly unique
  - can never eventually type with Π-E, or D-E, since those cannot type values in the empty ctx
- type-in-type, $\Xi \vdash v : \sigma$ is $\Xi \vdash \star : \star$
- Π-F, $\Xi \vdash v : \sigma$ is $\Xi \vdash \Pi x : \sigma.\tau\ : \star$
- D-F, $\Xi \vdash v : \sigma$ is $\Xi \vdash D\,\overline{M} : \star$
- D-F$'$, $\Xi \vdash v : \sigma$ is $\Xi \vdash D\,\overline{M} : \star$
- Π-I, $\Xi \vdash v : \sigma$ is $\Xi \vdash \mathsf{fun}\, f : (x.\tau)\,.\, x : \sigma.M\ :\ \mathit{\Pi} x : \sigma.\tau$
- D-I, $\Xi \vdash v : \sigma$ is $d\,\overline{N}\ :\ D\,\overline{M}'$ for some $\overline{M}'$
- no other typing rules are applicable

2.6 Progress

$\Xi \vdash M : \sigma$ implies that M is a value or there exists N such that $M \rightsquigarrow N$ and $\Xi \vdash \overline{M} : \Delta$ implies that $\overline{M}$ is a list of values or there exists $\overline{N}$ such that $\overline{M} \rightsquigarrow \overline{N}$

By mutual induction on the typing derivation and list typing derivation

Explicitly:

- M is typed by the conversion rule, then by **induction**, M is a value or there exists N such that $M \rightsquigarrow N$
- 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 there exist some σ, τ for $\Xi \vdash P : \Pi x : \sigma.\tau$ and $\Xi \vdash N : \sigma$. By **induction** (on the P branch of the derivation) P is a value or there exists P' such that $P \rightsquigarrow P'$. By **induction** (on the N branch of the derivation) N is a value or there exists N' such that $N \rightsquigarrow N'$
  - 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 $P\,N \rightsquigarrow P'\,N$
- M is typed by D-F'. M is $D\,\overline{N}$, a value
- M is typed by D-F. M is $D\,\overline{N}$, a value
- M is typed by D-I. By **induction** on lists
- M is typed by D-E. M is $\mathsf{Case}_{x:D\,\overline{y}.\sigma}\, N\, \mathsf{of} \left\{\overline{d_i\overline{x}_i \Rightarrow O_i\,|}\right\}$ By **induction** (on the N branch of the derivation) N is a value or there exists N' such that $N \rightsquigarrow N'$
  - if N is a value, by **canonical forms** N is $d_k\,\overline{v}$. from the typing derivation we know that there is a d_k clause in the case expression. The 1 step reduction is $\mathsf{Case}_{x:D\,\overline{y}.\sigma}\ (d_k\,\overline{v})\ \mathsf{of} \left\{\overline{d_i\overline{x}_i \Rightarrow O_i\,|}\right\} \rightsquigarrow O_k\,[\overline{x} := \overline{v}]$
  - otherwise, the one step reduction is $\mathsf{Case}_{x:D\,\overline{y}.\sigma}\, N\, \mathsf{of} \left\{\overline{d_i\overline{x}_i \Rightarrow O_i\,|}\right\} \rightsquigarrow \mathsf{Case}_{x:D\,\overline{y}.\sigma}\, N'\, \mathsf{of} \left\{\overline{d_i\overline{x}_i \Rightarrow O_i\,|}\right\}$
- $\overline{M}$ is typed by Δ-Trm-Emp. $\overline{M}$ is $\Diamond$ a degenerate value
- $\overline{M}$ is typed by Δ-Trm-$+$.
  - $\overline{M}$ is a list of values
  - $\overline{M}$ is $\overline{v}, N, \overline{N'}$. By **induction**

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

3 Conjectured properties

telescope regularity

$$\frac{\Gamma \vdash \overline{M} : \Delta}{\Gamma \vdash \Delta : \overline{\star}}$$

4 Non-Properties

- decidable type checking
- normalization/logical soundness

5 Differences from implementation

differences from Agda development

- In Agda presentation only handles functions, without data syntax
- In Agda the parallel reduction relation does not track the original typing judgment, though this should be equivalent
- Only proved the function part of the canonical forms lemma
- As here, standard properties are

differences from prototype

- bidirectional, type annotations are not always needed on functions or data
- top-level definitions of functions and data
  - top-level recursion function recursion is supported
  - top-level data references are supported
  - mutual recursion is allowed much more than in this presentation
- data constructors use re-use function application parameters rather then having a list of sub-terms
- aside from establishing that a term has a type, type annotations are not relevant for definitional equality in the prototype

6 Proof improvements

- abstract types subtly break the canonical forms lemma
- Clarify what unsoundness means
- proof outline at the top of document

- correct spelling
- better function notation
- meta syntax to quantify over contextual judgments
- meta syntax to quantify over $\Rrightarrow$ judgments
- clean up syntax
  - move to the more modern (a:M)-> N, instead of pi
  - clean up the case matching a bit
- make at terms represented by Latin characters, reserving the geek letters for other constructs
- enumerate the syntactic abbreviations
- Single (or double args) instead of the messy arg list notation