

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

February 11, 2021

1 Examples

logical unsoundness:

$\text{fun } f : (x.x) . x : \star . f x \quad : \Pi x : \star . x$

some constructs

while logically unsound, the language is extremely expressive. Conventional data is representable:

$\text{data } \mathbb{U} \text{. where } \{ \text{tt.} \rightarrow \mathbb{U} \}$

$\text{data } \mathbb{B} \text{. where } \{ \text{true.} \rightarrow \mathbb{B} \mid \text{false.} \rightarrow \mathbb{B} \}$

$\text{data } \mathbb{N} \text{. where } \{ z. \rightarrow \mathbb{N} \mid \text{suc } x : \mathbb{N}. \rightarrow \mathbb{N} \}$

data abstraction allows some self-reference

$0 := z$

$1 := \text{suc } z$

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

$\text{rep}_{\mathbb{B}} := \text{fun } f : (n.\text{Vec } \mathbb{B} \ n) . n : \mathbb{N} . \text{Case}_{n' : \mathbb{N} \rightarrow \text{Vec } \mathbb{B} \ n'} n \text{ of } \{ z \Rightarrow \text{nil } \mathbb{B} \mid \text{suc } x \Rightarrow \text{cons } \mathbb{B} \ x \text{ true } (f x) \}$:
 $\Pi n : \mathbb{N} . \text{Vec } \mathbb{B} \ n$

$\text{data } \text{Id } A : \star, a_1 : A, a_2 : A \text{. where } \{ \text{refl } A : \star, a : A. \rightarrow \text{Id } A \ a \ a \}$

$a_1 =_A a_2 := \text{Id } A, a_1, a_2$

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

$\text{subst} : \Pi A : \star . \Pi a_1 : A . \Pi a_2 : A . a_1 =_A a_2 \rightarrow \Pi C : (A \rightarrow \star) . C \ a_1 \rightarrow C \ a_2$

$\text{data } \perp \text{. where } \{ \}$

$$\neg A := \Pi A : \star..A \rightarrow \perp$$

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

$$\begin{aligned} dec &:= \lambda n. \mathbf{Case}_{-:\mathbb{N} \rightarrow \star} \mathbf{proof} \{z \Rightarrow \perp \mid suc - \Rightarrow Unit\} && : \mathbb{N} \rightarrow \star \\ \lambda pr. subst \mathbb{N} 1 0 pr dec tt && : \neg 1 =_{\mathbb{N}} 0 \end{aligned}$$

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:

$$\begin{aligned} &\frac{\Gamma, \Gamma' \vdash}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash M : \sigma}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash M \Rightarrow M' : \sigma}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash M \Rightarrow_* M' : \sigma}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash M \equiv M' : \sigma}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash \Delta : \bar{*}}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash \overline{M} : \Delta}{\Gamma \vdash} \\ &\frac{\Gamma, \Gamma' \vdash \overline{M} \Rightarrow \overline{M'} : \Delta}{\Gamma \vdash} \end{aligned}$$

by mutual induction on the derivations.

2.1.2 Context weakening

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

$$\begin{array}{c}
\frac{\Gamma, \Gamma' \vdash}{\Gamma, x : \sigma, \Gamma' \vdash} \\
\\
\frac{\Gamma, \Gamma' \vdash M : \tau}{\Gamma, x : \sigma, \Gamma' \vdash M : \tau} \\
\\
\frac{\Gamma, \Gamma' \vdash M \Rightarrow M' : \sigma}{\Gamma, x : \sigma, \Gamma' \vdash M \Rightarrow M' : \sigma} \\
\\
\frac{\Gamma, \Gamma' \vdash M \Rightarrow_* M' : \sigma}{\Gamma, x : \sigma, \Gamma' \vdash M \Rightarrow_* M' : \sigma} \\
\\
\frac{\Gamma, \Gamma' \vdash M \equiv M' : \tau}{\Gamma, x : \sigma, \Gamma' \vdash M \equiv M' : \tau} \\
\\
\frac{\Gamma, \Gamma' \vdash \Delta : \bar{*}}{\Gamma, x : \sigma, \Gamma' \vdash \Delta : \bar{*}} \\
\\
\frac{\Gamma, \Gamma' \vdash \overline{M} : \Delta}{\Gamma, x : \sigma, \Gamma' \vdash \overline{M} : \Delta} \\
\\
\frac{\Gamma, \Gamma' \vdash \overline{M} \Rightarrow \overline{M'} : \Delta}{\Gamma, x : \sigma, \Gamma' \vdash \overline{M} \Rightarrow \overline{M'} : \Delta}
\end{array}$$

by mutual induction on the derivations.

2.1.3 \Rightarrow is reflexive

The following rule is admissible:

$$\begin{array}{c}
\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M \Rightarrow M : \sigma} \Rightarrow\text{-refl} \\
\\
\frac{\Gamma \vdash \overline{M} : \Delta}{\Gamma \vdash \overline{M} \Rightarrow \overline{M} : \Delta} \Rightarrow\text{-refl}'
\end{array}$$

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 \Rightarrow *-refl

2.1.5 Context substitution

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

$$\begin{array}{c}
\frac{\Gamma, x : \tau, \Gamma' \vdash}{\Gamma, \Gamma' [x := N] \vdash} \\
\\
\frac{\Gamma, x : \tau, \Gamma' \vdash M : \sigma}{\Gamma, \Gamma' [x := N] \vdash M [x := N] : \sigma [x := N]} \\
\\
\frac{\Gamma, x : \tau, \Gamma' \vdash M \Rightarrow M' : \sigma}{\Gamma, \Gamma' [x := N] \vdash M [x := N] \Rightarrow M' [x := N] : \sigma [x := N]} \\
\\
\frac{\Gamma, x : \tau, \Gamma' \vdash M \Rightarrow_* M' : \sigma}{\Gamma, \Gamma' [x := N] \vdash M [x := N] \Rightarrow_* M' [x := N] : \sigma} \\
\\
\frac{\Gamma, x : \tau, \Gamma' \vdash M \equiv M' : \sigma}{\Gamma, \Gamma' [x := N] \vdash M [x := N] \equiv M' [x := N] : \sigma [x := N]} \\
\\
\frac{\Gamma, x : \tau, \Gamma' \vdash \Delta : \bar{*}}{\Gamma, \Gamma' [x := N] \vdash \Delta [x := N] : \bar{*}} \\
\\
\frac{\Gamma, x : \tau, \Gamma' \vdash \bar{M} : \Delta}{\Gamma, \Gamma' [x := N] \vdash \bar{M} [x := N] : \Delta [x := N]} \\
\\
\frac{\Gamma, x : \tau, \bar{M} \Rightarrow \bar{M}' : \Delta}{\Gamma, \Gamma' [x := N] \vdash \bar{M} [x := N] \Rightarrow \bar{M}' [x := N] : \Delta [x := N]}
\end{array}$$

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 $\Gamma, \Gamma' [x := N] \vdash N : \tau$, and apply appropriate reflexivity when needed.

2.2 Computation

2.2.1 \Rightarrow preserves type of source

The following rules are admissible:

$$\begin{array}{c}
\frac{\Gamma \vdash N \Rightarrow N' : \tau}{\Gamma \vdash N : \tau} \\
\\
\frac{\Gamma \vdash \bar{N} \Rightarrow \bar{N}' : \Delta}{\Gamma \vdash \bar{N} : \Delta}
\end{array}$$

by mutual induction

2.2.2 \Rightarrow substitution

The following rule is admissible:

$$\frac{\Gamma, \Delta, \Gamma' \vdash \overline{M} \Rightarrow \overline{M'} : \Theta \quad \Gamma \vdash \overline{N} \Rightarrow \overline{N'} : \Delta}{\Gamma, \Gamma' [\Delta := \overline{N}] \vdash \overline{M} [\Delta := \overline{N}] \Rightarrow \overline{M'} [\Delta := \overline{N'}] : \Theta [\Delta := \overline{N}]}$$

by induction on the \Rightarrow derivations with the corollary

Corollary, the following rule is admissible:

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

2.2.3 \Rightarrow is confluent

if $\Gamma \vdash M \Rightarrow N : \sigma$ and $\Gamma \vdash M \Rightarrow N' : \sigma$ then there exists P such that $\Gamma \vdash N \Rightarrow P : \sigma$ and $\Gamma \vdash N' \Rightarrow P : \sigma$

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

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

- Π -E- \Rightarrow and Π - \Rightarrow

- M is $(\text{fun } f : (x.\tau) . x : \sigma.B) A$
- N is $(\text{fun } f : (x.\tau) . x : \sigma.B') A' , B \Rightarrow B' , A \Rightarrow A'$
- N' is $B'' [x := A''] , f := (\text{fun } f : (x.\tau) . x : \sigma.B'') , B \Rightarrow B'' , A \Rightarrow A''$
- $B \Rightarrow B_v , A \Rightarrow A_v$ by I.H
- $(\text{fun } f : (x.\tau) . x : \sigma.B) A \Rightarrow B_v [x := A_v, f := (\text{fun } f : (x.\tau) . x : \sigma.B_v)]$
by repeated \Rightarrow substitution

- D -E- \Rightarrow and D - \Rightarrow

- M is $\text{Case}_{x:D \overline{y}.\sigma} (d \overline{A})$ of $\left\{ \overline{d_i \overline{x}_i} \Rightarrow \overline{B_i} \right\}$
- N is $\text{Case}_{x:D \overline{y}.\sigma} (d \overline{A'})$ of $\left\{ \overline{d_i \overline{x}_i} \Rightarrow \overline{B'_i} \right\}, \overline{A} \Rightarrow \overline{A'}, \forall i. B_i \Rightarrow B'_i$
- N' is $B'' [\overline{x} := \overline{A''}], \overline{A} \Rightarrow \overline{A''}, B \Rightarrow B'', d\overline{x} \Rightarrow B \in_i \left\{ \overline{d_i \overline{x}_i} \Rightarrow \overline{B_i} \right\}$
- $B \Rightarrow B_v, \overline{A} \Rightarrow \overline{A_v}$ by I.H
- $\text{Case}_{x:D \overline{y}.\sigma} (d \overline{A})$ of $\left\{ \overline{d_i \overline{x}_i} \Rightarrow \overline{B_i} \right\} \Rightarrow B_v [\overline{x} := \overline{A_v}]$ by repeated \Rightarrow substitution

- all other reductions match, and follow immediately from induction, or are symmetric to already presented cases

2.3 \Rightarrow_*

2.3.1 \Rightarrow_* is transitive

The following rule is admissible:

$$\frac{\Gamma \vdash M \Rightarrow_* M' : \sigma \quad \Gamma \vdash M' \Rightarrow_* M'' : \sigma}{\Gamma \vdash M \Rightarrow_* M'' : \sigma} \Rightarrow \text{*trans}$$

by induction

2.3.2 \Rightarrow preserves type in source

The following rules are admissible:

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

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

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

- $\Pi\text{-}\Rightarrow$

- $M' [x := N', f := (\text{fun } f : (x.\tau) . x : \sigma.M')] : \tau [x := N]$ by the substitution lemma used on the inductive hypotheses

- $D\text{-}\Rightarrow$

- $O' [\overline{x} := \overline{N'}] : \sigma [x := d\overline{x}_i, \overline{y} := \overline{N}]$ by the substitution lemma used on the inductive hypotheses

- all other cases are trivial

2.3.3 \Rightarrow_* preserves type

The following rule is admissible:

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

by induction

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

by induction

2.3.4 \Rightarrow_* is confluent

if $\Gamma \vdash M \Rightarrow_* N : \sigma$ and $\Gamma \vdash M \Rightarrow_* N' : \sigma$ then there exists P such that

$$\Gamma \vdash N \Rightarrow_* P : \sigma \text{ and } \Gamma \vdash N' \Rightarrow_* P : \sigma$$

Follows from \Rightarrow *-trans and the confluence of \Rightarrow 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

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

by the confluence of \Rightarrow_*

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 \Rightarrow_*

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 \Rightarrow

The following rules are admissible:

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

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

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 \Rightarrow and \Rightarrow preserves types

2.4 Type constructors

2.4.1 Type constructors are stable

- if $\Gamma \vdash * \Rightarrow M : \sigma$ then M is $*$
- if $\Gamma \vdash * \Rightarrow_* M : \sigma$ then M is $*$
- if $\Gamma \vdash \Pi x : \sigma.\tau \Rightarrow M : \sigma$ then M is $\Pi x : \sigma'.\tau'$ for some σ', τ'
- if $\Gamma \vdash \Pi x : \sigma.\tau \Rightarrow_* M : \sigma$ then M is $\Pi x : \sigma'.\tau'$ for some σ', τ'
- if $\Gamma \vdash D \overline{M} \Rightarrow M : \sigma$ then M is $D \overline{M'}$ for some $\overline{M'}$
- if $\Gamma \vdash D \overline{M} \Rightarrow_* M : \sigma$ then M is $D \overline{M'}$ for some $\overline{M'}$

by induction on the respective relations

2.4.2 Type constructors definitionally 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 $\text{fun } f : (x.\tau') . x : \sigma''.P'$ for some τ', σ'', P'
- if σ is $D \overline{M}$ for some \overline{M} then v is $d \overline{v}$

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 definitionally 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, Π -F, D -F, D -F', or D -I since constructors are definitionally unique
 - if σ is $D \overline{M}$ then eventually, it was typed with D -I. it could not have been typed by type-in-type, Π -F, D -F, D -F', or Π -I since constructors are definitionally 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 $\Xi \vdash \text{fun } f : (x.\tau).x : \sigma.M : \Pi x : \sigma.\tau$
- 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 $\text{fun } f : (x.\tau).x : \sigma.M'$, 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 $\text{fun } f : (x.\tau) . x : \sigma.P'$ and
 - * if N is a value then the one step reduction is $(\text{fun } f : (x.\tau) . x : \sigma.P') N \rightsquigarrow P'[x := N, f := \text{fun } f : (x.\tau) . x : \sigma.M]$
 - * otherwise there exists N' such that $N \rightsquigarrow N'$, and the one step reduction is $(\text{fun } f : (x.\tau) . x : \sigma.P') N \rightsquigarrow (\text{fun } f : (x.\tau) . x : \sigma.P') N'$
 - otherwise, there exists P' such that $P \rightsquigarrow P'$ and the one step reduction is $P N \rightsquigarrow P' N$
- M is typed by $D\text{-F'}$. M is $D\overline{N}$, a value
 - M is typed by $D\text{-F}$. M is $D\overline{N}$, a value
 - M is typed by $D\text{-I}$. By **induction** on lists
 - M is typed by $D\text{-E}$. M is $\text{Case}_{x:D \ \overline{y}.\sigma} N$ of $\{\overline{d_i \overline{x}_i \Rightarrow O_i} \mid\}$ 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 $d\overline{v}$. from the typing derivation we know that there is a d clause in the case expression. The 1 step reduction is $\text{Case}_{x:D \ \overline{y}.\sigma} (D\overline{v})$ of $\{\overline{d_i \overline{x}_i \Rightarrow O_i} \mid\} \rightsquigarrow O[\overline{x} := \overline{v}]$
 - otherwise, the one step reduction is $\text{Case}_{x:D \ \overline{y}.\sigma} N$ of $\{\overline{d_i \overline{x}_i \Rightarrow O_i} \mid\} \rightsquigarrow \text{Case}_{x:D \ \overline{y}.\sigma} N'$ of $\{\overline{d_i \overline{x}_i \Rightarrow O_i} \mid\}$
 - \overline{M} is typed by $\Delta\text{-Trm-Emp}$. \overline{M} is \diamond a degenerate value
 - \overline{M} is typed by $\Delta\text{-Trm-+}$.
 - \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 than having a list of subterms
- aside from establishing that a term has a type, type annotations are not relevant for definitional equality in the prototype

6 Proof improvements

- meta syntax to quantify over contextual judgments
- meta syntax to quantify over \Rightarrow judgments
- clean up syntax
 - move to the more modern $(a:M) \rightarrow N$, instead of π
 - 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