Interacting safely with an unsafe environment

Gilles Dowek
Summary of the talk

1985: Coquand and Huet gave a presentation of the Calculus of constructions: 8 rules
1991: Geuvers and Nederhof simplified this presentation: 7 rules
2021: a simpler presentation: 6 rules
The main idea

Forget the idea that contexts should be well-formed
In simply typed $\lambda$-calculus

Assign $nat \rightarrow nat \rightarrow nat$ to $f$ and $nat$ to $x$
Then $\lambda y : nat \ (f \ x \ y)$ has type $nat \rightarrow nat$

Whether a type is assigned to $f$ before or after $x$ immaterial
Thus, no order in the context $\{f : nat \rightarrow nat \rightarrow nat, x : nat\}$
When atomic types are variables of type $\ast$

\[
\lambda y : \text{nat} \ (f \times y) \text{ has type } \text{nat} \to \text{nat}
\]

in the context $\text{nat} : \ast, f : \text{nat} \to \text{nat} \to \text{nat}, x : \text{nat}$

but not in $x : \text{nat}, \text{nat} : \ast, f : \text{nat} \to \text{nat} \to \text{nat}$

Not well-formed

- The type of a variable contains variables declared to its left
- Each type is well-typed in the context formed with the variable to its left

$\text{nat} : \ast, \text{array} : \text{nat} \to \ast, z : \text{nat}, \text{nil} : (\text{array} z z)$ not well-formed
The original formulation of the Calculus of Constructions

Two forms of judgements: $\Gamma \vdash t : A$ and $\Gamma$ well-formed

Two rules define when a context is well-formed

\[
\begin{align*}
\text{[ ] well-formed } & \quad \text{(empty)} \\
\Gamma \vdash A : s & \quad \text{(decl) } s \in \{*, \Box\} \\
\Gamma, x : A & \quad \text{well-formed (decl) } s \in \{*, \Box\} \\
\Gamma, x : A, \Gamma' & \quad \text{well-formed (var)} \\
\end{align*}
\]

and a rule (var)

\[
\begin{align*}
\Gamma, x : A, \Gamma' \quad \text{well-formed} & \quad \text{(var)} \\
\Gamma, x : A, \Gamma' \vdash x : A & \\
\end{align*}
\]

5 others: (sort), (prod), (abs), (app), and (conv): 8 rules

Because of (var), a variable can only be assigned a type in a well-formed context (extends to all terms, invariant)
The formulation of Geuvers and Nederhof

- Drop the context $\Gamma'$ in (var)

$$\frac{\text{$\Gamma, x : A$ well-formed}}{\text{$\Gamma, x : A \vdash x : A$}}$$

- Add a weakening rule

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s}{\Gamma, x : B \vdash t : A} \quad \text{(weak)}$$

Then conclusion of (decl) = premise of (var) ($\Gamma, x : A$ well-formed)

Coin a derived rule

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \quad \text{(start)} \ s \in \{ \ast, \Box \}$$

Drop $\Gamma$ well-formed and (empty), (decl), and (var) (start) + (weak) + the 5 other rules = 7 rules

All contexts well-formed (invariant)
A derivation of $\Gamma \vdash x : A$

The well-typedness of the term $A$ needs to be checked:

- when the variable $x$ is added to the context
- when it is used in the derivation of $\Gamma \vdash x : A$

(\text{empty}) + (\text{decl}) + (\text{var}) when it is added to the context
(\text{weak}) + (\text{start}) when the variable is used

Two approaches to safety: build a safe environment, interact safely with a possibly unsafe environment
Yet

In Geuvers and Nederhof contexts are always well-formed

\[
\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s}{\Gamma, x : B \vdash t : A} \text{ (weak)}
\]

instrumental (to preserve the well-formedness of the context)
One step further

\[
\frac{\Gamma \vdash t : A}{\Gamma, x : B \vdash t : A}
\]

- In $\Gamma \vdash x : A$, $\Gamma$ needs not be well-formed
- but $A$ still well-typed (rule (start))

\[
\text{nat} : *, \text{array} : \text{nat} \to *, z : \text{nat}, \text{nil} : (\text{array} z z) \vdash z : \text{nat} \text{ derivable}
\]

But not

\[
\text{nat} : *, \text{array} : \text{nat} \to *, z : \text{nat}, \text{nil} : (\text{array} z z) \vdash \text{nil} : (\text{array} z z)
\]
Dropping the weakening rule

Extend the (start) rule to

\[
\frac{\Gamma, x : A, \Gamma' \vdash A : s}{\Gamma, x : A, \Gamma' \vdash x : A} \text{ (var')} 
\]

- adding $\Gamma'$ permits to drop the weakening rule
- adding $\Gamma'$ makes the order of declarations (sets or lists) immaterial

$(\text{var'}) +$ the 5 other rules $= 6$ rules
Why arbitrary contexts?

- When merging $\Gamma$ and $\Gamma'$, developed by different teams in different places no choice between $\Gamma, \Gamma'$, and $\Gamma', \Gamma$: unordered context $\Gamma \cup \Gamma'$ (Sacerdoti Coen)

- When we extend type systems with rewrite rules
  - the rules $\mathcal{R}$ use the symbols declared in $\Gamma$ ($\Gamma$ before)
  - the well-formedness of $\Gamma$ requires the rules of $\mathcal{R}$ ($\mathcal{R}$ before) (Blanqui)

For example: $\Gamma = nat : *, a : nat, b : nat, P : nat \to *, Q : (P a) \to *, e : (P b), h : (Q e), c : nat$

$\mathcal{R} = a \rightarrow c, b \rightarrow c$

- (only third) a way to minimize the number of rules
Tribute

- Coquand and Huet, Geuvers and Nederhof, Sacerdoti Coen, Blanqui

- But also Geuvers, Krebbers, McKinna, and Wiedijk who (wilder) completely dropped contexts
  Here: no modification of the syntax of terms
  No issue whether $x^B \equiv x^{(\lambda A:*A)} B$ or not
Lemma 1 (Key lemma)
If

- $\Gamma \vdash t : A$ derivable \textit{in the new system}
- and $\Gamma$ well-formed (in the usual one)

then $\Gamma \vdash t : A$ derivable \textit{in the usual one}
Two lemmas

Lemma 2 (Context curation)
If \( \Gamma \vdash t : A \) derivable in the new system, then there exists \( \Delta \) such that

\[
\begin{align*}
\Delta & \text{ well-formed (in the usual one)} \\
\Delta & \subseteq \Gamma \\
\Delta & \vdash t : A \text{ is derivable in the new one}
\end{align*}
\]

Because of (var’), the structure of a derivation tree induces a partial order between the used variables of \( \Gamma \)
Topological sorting of the used variables yields \( \Delta \)
And a theorem

If $\Gamma \vdash t : A$ derivable in the new system, then there exists $\Delta$, such that

1. $\Delta \subseteq \Gamma$
2. $\Delta \vdash t : A$ derivable in the usual one
Well-formedness of contexts is a useful property

But it should not be part of the definition of type systems: simpler, more convenient, and easier to extend systems