Interacting Safely with an Unsafe Environment

Gilles Dowek

Interacting Safely with an Unsafe Environment

Gilles Dowek
Inria and ENS Paris-Saclay
gilles.dowek@ens-paris-saclay.fr

We give a presentation of Pure type systems where contexts need not be well-formed and show that this presentation is equivalent to the usual one. The main motivation for this presentation is that, when we extend Pure type systems with computation rules, like in the logical framework DEDUKTI, we want to declare the constants before the computation rules that are needed to check the well-typedness of their type.

1 Introduction

In the simply typed lambda-calculus, to assign a type to a term, we first need to assign a type to its free variables. For instance, if we assign the type $nat \to nat \to nat$ to the variable f and the type nat to the variable x, then we can assign the type $nat \to nat$ to the term $\lambda y : nat\ (f\ x\ y)$.

Whether a type is assigned to f before or after one is assigned to x is immaterial, so the context $\{f : nat \to nat \to nat, x : nat\}$ does not need to be ordered.

1.1 Well-formed Contexts

In systems, such as the Calculus of constructions, where atomic types are variables of a special type $*$, contexts are ordered and, for instance, the term $\lambda y : nat\ (f\ x\ y)$ is assigned the type $nat \to nat$ in the context $nat : *, f : nat \to nat \to nat, x : nat$ but not in the context $x : nat, nat : *, f : nat \to nat \to nat$, that is not well-formed.

In a well-formed context, the declarations are ordered in such a way that the type of a variable only contains variables declared to its left. For instance, the context $nat : *, z : nat, array : nat \to *, nil : (array\ z)$ is well-formed, but the context $nat : *, z : nat, nil : (array\ z), array : nat \to *$ is not. Moreover, in such a well-formed context, each type is itself well-typed in the context formed with the variable declarations to its left. For instance, the context $nat : *, array : nat \to *, z : nat, nil : (array\ z\ z)$ is not well-formed. So, a context $x_1 : A_1, ..., x_n : A_n$ is said to be well-formed if, for each i, $x_1 : A_1, ..., x_i : A_i \vdash A_{i+1} : s$ is derivable for some sort s in $\{*, \Box\}$.

The original formulation of the Calculus of constructions of Coquand and Huet [4] has two forms of judgements: one expressing that a context Γ is well-formed and another expressing that a term t has a type A in a context Γ. Two rules define when a context is well-formed

$$\frac{}{[\,]\ \text{well-formed}}\ (\text{empty})$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A\ \text{well-formed}}\ (\text{decl})\ s \in \{*, \Box\}$$

and one enables the assignment of a type to a variable, in a well-formed context

$$\frac{\Gamma, x : A, \Gamma'\ \text{well-formed}}{\Gamma, x : A, \Gamma' \vdash x : A}\ (\text{var})$$

E. Pimentel and E. Tassi (Eds.): Logical Frameworks
and Meta-Languages: Theory and Practice (LFMTP 2021)
EPTCS 337, 2021, pp. 30–38, doi:10.4204/EPTCS.337.3

These three rules together with five others—(sort), (prod), (abs), (app), and (conv)—form a eight-rule presentation of the Calculus of constructions, and more generally of Pure type systems. Because the rule (var) requires the context $\Gamma, x : A, \Gamma'$ to be well-formed, a variable can only be assigned a type in a well-formed context and this property extends to all terms, as it is an invariant of the typing rules.

This system was simplified by Geuvers and Nederhof [7] and Barendregt [1], who use a single form of judgement expressing that a term t has a type A in a context Γ. First, they drop the context Γ' in the rule (var) simplifying it to

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

and add a weakening rule

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

to extend the judgement $\Gamma, x : A \vdash x : A$ to $\Gamma, x : A, \Gamma' \vdash x : A$. Then, they exploit the fact that the conclusion of the rule (decl) is now identical to the premise of the rule (var), to coin a derived rule

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

Now that the variables can be typed without using a judgement of the form Γ well-formed, such judgements can be dropped, together with the rules (empty), (decl), and (var). So the two rules (start) and (weak), together with the five other rules form an equivalent seven-rule formulation of the Calculus of constructions, and more generally of Pure type systems.

1.2 Interacting Safely with an Unsafe Environment

When a judgement of the form $\Gamma \vdash x : A$ is derived, the well-typedness of the term A needs to be checked. But it can be checked either when the variable x is added to the context or when it is used in the derivation of the judgement $\Gamma \vdash x : A$. In the system with the rules (decl) and (var), it is checked in the rule (decl), that is when the variable is added to the context. When the rule (var) is replaced with the rule (start), it is checked when the variable is used. These two systems illustrate two approaches to safety: the first is to build a safe environment, the second is to interact safely with a possibly unsafe environment.

In the formulation of Geuvers and Nederhof and Barendregt, it is still possible to define a notion of well-formed context: the context $x_1 : A_1, ..., x_n : A_n$ is well-formed if for each i, $x_1 : A_1, ..., x_i : A_i \vdash A_{i+1} : s$ is derivable. With such a definition, it is possible to prove that if the judgement $\Gamma \vdash t : A$ is derivable, then Γ is well-formed. In this proof, the second premise of the rule (weak), $\Gamma \vdash B : s$, is instrumental, as its only purpose is to preserve the well-formedness of the context.

We can go further with the idea of interacting safely with an unsafe environment and drop this second premise, leading to the weakening rule

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

Then, in the judgement $\Gamma \vdash x : A$, nothing prevents the context Γ from being non well-formed, but the term A is still well-typed because the rule (start), unlike the rule (var), has a premise $\Gamma \vdash A : s$. In such a system, the judgement $nat : *, array : nat \to *, z : nat, nil : (array\ z\ z) \vdash z : nat$ is derivable, although the term $(array\ z\ z)$ is not well-typed, but the judgement $nat : *, array : nat \to *, z : nat, nil : (array\ z\ z) \vdash nil : (array\ z\ z)$ is not because this term $(array\ z\ z)$ is not well-typed.

$$\frac{}{\Gamma \vdash s_1 : s_2}\,(\text{sort'}) \quad \langle s_1, s_2\rangle \in \mathscr{A}$$

$$\frac{\Gamma, x : A, \Gamma' \vdash A : s}{\Gamma, x : A, \Gamma' \vdash x : A}\,(\text{var'}) \quad x \in \mathscr{V}_s$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (x : A) \to B : s_3}\,(\text{prod}) \quad \langle s_1, s_2, s_3\rangle \in \mathscr{R}$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : (x : A) \to B}\,(\text{abs}) \quad \langle s_1, s_2, s_3\rangle \in \mathscr{R}$$

$$\frac{\Gamma \vdash t : (x : A) \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : (u/x)B}\,(\text{app})$$

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

Figure 1: Pure type systems with arbitrary contexts

Yet, with the rule (start) and this strong weakening rule, the judgement $nat : *, z : nat, nil : (array\ z)$, $array : nat \to * \vdash nil : (array\ z)$ is not derivable, because the judgement $nat : *, z : nat \vdash (array\ z) : *$ is not derivable. Thus, to make this judgement derivable, we should not use a weakening rule that erases all the declarations to the right of the declaration of *nil* and the rule (start). But we should instead use a rule that keeps the full context to type the term $(array\ z)$. Yet, like the rule (start), this rule should not check that the context is well-formed, but that the type of the variable is a well-typed term

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

This leads to the six-rule system described in Figure 1.

As the order of declarations in a context is now immaterial, contexts can indifferently be defined as sequences or as sets of declarations.

1.3 Previous Work

There are several reasons for using arbitrary contexts. One of them is that, as already noticed by Sacerdoti Coen [3], when we have two contexts Γ and Γ', for instance developed by different teams in different places, and we want to merge them, we should not have to make a choice between Γ, Γ', and Γ', Γ. We should just be able to consider the unordered context $\Gamma \cup \Gamma'$, provided it is a context, that is if $x : A$ is declared in Γ and $x : A'$ is declared in Γ' then $A = A'$.

Another is that, when we extend Pure type systems with computation rules, like in the logical framework DEDUKTI, we additionally want to declare constants in a signature Σ and then add computation rules. For instance, we want to be able to declare constants in a signature $\Sigma = nat : *, a : nat, b : nat, P : nat \to *, Q : (P\ a) \to *, e : (P\ b), h : (Q\ e), c : nat$ and then computation rules $a \longrightarrow c$, $b \longrightarrow c$. Because, unlike in [5], the term $(Q\ e)$ is not well-typed without the computation rules, we cannot check the that the signature is well-formed before we declare the rules. But, because the rules use the constants declared in Σ, we cannot declare the rules before the signature, in particular the rules do not make sense in the part of the signature to the left of the declaration of h, that is in $nat : *, a : nat, b : nat, P : nat \to *, Q : (P\ a) \to$

$*, e : (P\ b)$, where the constant c is missing. And, because we sometimes want to consider rules $l \longrightarrow r$ where l and r are not well-typed terms [2], we cannot interleave constant declarations and computation rules. Note that in Blanqui's Calculus of algebraic constructions [2], the contexts are required to be well-formed, but the signatures are not.

Another source of inspiration is the presentation of Pure type systems without explicit contexts [6], where Geuvers, Krebbers, McKinna, and Wiedijk completely drop contexts in the presentation of Pure type systems. In particular, Theorem 3.1 below is similar to their Theorem 19. The presentation of Figure 1 is however milder than their Pure type systems without explicit contexts, as it does not change the syntax of terms, avoiding, for instance, the question of the convertibility of x^B and $x^{(\lambda \dot{A}:*\dot{A})\ B}$. In particular, if $\Gamma \vdash t : A$ is derivable in the usual formulation of Pure type systems, it is also derivable in the system of Figure 1.

We show, in this note, that the system presented in Figure 1 indeed allows to interact safely with an unsafe environment, in the sense that if a judgement $\Gamma \vdash t : A$ is derivable in this system, then there exists Δ, such that $\Delta \subseteq \Gamma$ and $\Delta \vdash t : A$ is derivable with the usual Pure type system rules. The intuition is that, because of the rule (var'), the structure of a derivation tree induces a dependency between the used variables of Γ that is a partial order, and as already noticed by Sacerdoti Coen [3], a topological sorting of the used variables yields a linear context Δ. Topological sorting is the key of Lemma 3.4.

So this paper build upon the work of Coquand and Huet [4], Geuvers and Nederhof [7], Barendregt [1], Blanqui [2], Sacerdoti Coen [3], and Geuvers, Krebbers, McKinna, and Wiedijk [6]. Its main contribution is to show that Pure type systems can be defined with six rules only, without a primitive notion of well-formed context, and without changing the syntax of terms.

2 Pure Type Systems

Let us first recall a usual definition of (functional) Pure type systems [7, 1]. To define the syntax of terms, we consider a set $\mathscr{S}$ of sorts and a family of $\mathscr{V}_s$ of infinite and disjoint sets of variables of sort s. The syntax is then

$$t = x \mid s \mid (x : A) \to B \mid \lambda x : A\ t \mid t\ u$$

A context Γ is a sequence $x_1 : A_1, ..., x_n : A_n$ of pairs formed with a variable and a term, such that the variables $x_1, ..., x_n$ are distinct. So, when we write the context $\Gamma, y : B$, we implicitly assume that y is not already declared in Γ.

A context Γ is said to be included into a context Γ' ($\Gamma \subseteq \Gamma'$) if every $x : A$ in Γ is also in Γ'.

Two contexts Γ and Γ' are said to be compatible if each time $x : A$ is in Γ and $x : A'$ is in Γ', then $A = A'$.

To define the typing rule, we consider a set $\mathscr{A}$ of axioms, that are pairs of sorts and a set $\mathscr{R}$ of rules, that are triple of sorts. As we restrict to functional Pure type systems, we assume that the relations $\mathscr{A}$ and $\mathscr{R}$ are functional.

Definition 2.1 (The type system $\mathscr{T}$)

$$\frac{}{\vdash s_1 : s_2}\,(sort) \quad \langle s_1, s_2 \rangle \in \mathscr{A}$$

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\,(start) \quad x \in \mathscr{V}_s$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s}{\Gamma, x : B \vdash t : A}\,(weak) \quad x \in \mathscr{V}_s$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash (x : A) \to B : s_3}\,(prod) \quad \langle s_1, s_2, s_3 \rangle \in \mathscr{R}$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : (x : A) \to B}\,(abs) \quad \langle s_1, s_2, s_3 \rangle \in \mathscr{R}$$

$$\frac{\Gamma \vdash t : (x : A) \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : (u/x)B}\,(app)$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s}{\Gamma \vdash t : B}\,(conv) \quad A \equiv B$$

Example. Consider two sorts $*$ and $\Box$ and an axiom $* : \Box$. The judgement $nat : *, z : nat \vdash z : nat$ is derivable in $\mathscr{T}$. But the judgements $z : nat, nat : * \vdash z : nat$ is not because z is declared before *nat* and the judgement $nat : *, x : (* *), z : nat \vdash z : nat$ is not because $(* *)$ is not well-typed.

Definition 2.2 (Well-formed) *Well-formed contexts are inductively defined with the rules*

- *the empty context is well-formed,*
- *if* Γ *is well-formed and* $\Gamma \vdash A : s$ *is derivable in* $\mathscr{T}$*, then* $\Gamma, x : A$ *is well-formed.*

Lemma 2.1 *If* $\Gamma \vdash t : A$ *is derivable, then* Γ *is well-formed. Conversely, if* Γ *is well-formed, then there exist two terms t and A, such that* $\Gamma \vdash t : A$ *is derivable.*

Proof. We prove that Γ is well-formed, by induction on the derivation of $\Gamma \vdash t : A$. Conversely, if Γ is well-formed and s_1 and s_2 are two sorts, such that $\langle s_1, s_2 \rangle \in \mathscr{A}$ then $\Gamma \vdash s_1 : s_2$ is derivable with the rules (sort) and (weak).

We will use the two following lemmas. The first is Lemma 18 in [7] and 5.2.12 in [1] and the second Lemma 26 in [7] and 5.2.17 in [1].

Lemma 2.2 (Thinning) *If* $\Gamma \vdash t : A$ *is derivable,* $\Gamma \subseteq \Gamma'$*, and* Γ' *is well-formed, then* $\Gamma' \vdash t : A$ *is derivable.*

Lemma 2.3 (Strengthening) *If* $\Gamma, x : A, \Gamma' \vdash t : B$ *is derivable and x does not occur in* Γ'*, t, and A, then* $\Gamma, \Gamma' \vdash t : B$ *is derivable.*

Lemma 2.4 (Strengthening contexts) *If* $\Gamma_1, x : A, \Gamma_2$ *is well-formed and x does not occur in* Γ_2 *then* Γ_1, Γ_2 *is well-formed.*

Proof. By induction on the structure of Γ_2. If Γ_2 is empty, then $\Gamma_1, \Gamma_2 = \Gamma_1$ is well-formed. Otherwise, $\Gamma_2 = \Gamma_2', y : B$. By induction hypothesis, Γ_1, Γ_2' is well-formed. As $\Gamma_1, x : A, \Gamma_2', y : B$ is well-formed, $\Gamma_1, x : A, \Gamma_2' \vdash B : s$ is derivable. By Lemma 2.3, $\Gamma_1, \Gamma_2' \vdash B : s$ is derivable. Thus, $\Gamma_1, \Gamma_2', y : B$ is well-formed.

If Γ_1 and Γ_2 are two well-formed contexts with no variables in common, then the concatenation Γ_1, Γ_2 also is well-formed. This remark extend to the case where Γ_1 and Γ_2 have variables in common, but are compatible.

Lemma 2.5 (Merging) *If* Γ_1 *and* Γ_2 *are two well-formed compatible contexts, then there exists a well-formed context* Γ*, such that* $\Gamma_1 \subseteq \Gamma$*,* $\Gamma_2 \subseteq \Gamma$*, and* $\Gamma \subseteq (\Gamma_1, \Gamma_2)$.

Proof. By induction on of Γ_2.

- If Γ_2 is empty, we take $\Gamma = \Gamma_1$. The context Γ is well-formed, $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, and $\Gamma \subseteq (\Gamma_1, \Gamma_2)$.

- If $\Gamma_2 = (\Gamma'_2, x : A)$, then Γ'_2 is well-formed and, by induction hypothesis, there exists a well-formed context Γ', such that $\Gamma_1 \subseteq \Gamma'$, $\Gamma'_2 \subseteq \Gamma'$, and $\Gamma' \subseteq (\Gamma_1, \Gamma'_2)$.
  - If $x : A \in \Gamma'$, then we take $\Gamma = \Gamma'$. The context Γ is well-formed, $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, and $\Gamma \subseteq (\Gamma_1, \Gamma_2)$.
  - Otherwise, as Γ_1 and Γ_2 are compatible, Γ' contains no other declaration of x. We take $\Gamma = \Gamma', x : A$. We have $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, $\Gamma \subseteq (\Gamma_1, \Gamma_2)$. By Lemma 2.2, as $\Gamma'_2 \vdash A : s$, and $\Gamma'_2 \subseteq \Gamma'$, and Γ' is well-formed, $\Gamma' \vdash A : s$ is derivable , thus Γ is well-formed.

Example. Consider two sorts $*$ and $\Box$ and an axiom $* : \Box$. If $\Gamma_1 = \mathit{nat} : *, \mathit{bool} : *, z : \mathit{nat}$ and $\Gamma_2 = \mathit{bool} : *, \mathit{true} : \mathit{bool}, \mathit{nat} : *$, the context Γ is $\mathit{nat} : *, \mathit{bool} : *, z : \mathit{nat}, \mathit{true} : \mathit{bool}$.

3 Arbitrary Contexts

Definition 3.1 (The type system $\mathscr{T}'$) *The system $\mathscr{T}'$ is formed with the rules of Figure 1. With respect to the system $\mathscr{T}$, the rule (sort) is replaced with the rule (sort'), the rule (start) is replaced with the rule (var'), and the rule (weak) is dropped.*

Example. Consider two sorts $*$ and $\Box$ and an axiom $* : \Box$. The judgement $\mathit{nat} : *, z : \mathit{nat} \vdash z : \mathit{nat}$ is derivable in $\mathscr{T}'$. So are the judgements $z : \mathit{nat}, \mathit{nat} : * \vdash z : \mathit{nat}$ and $\mathit{nat} : *, x : (* *), z : \mathit{nat} \vdash z : \mathit{nat}$.

Lemma 3.1 (Thinning) *If Γ and Γ' are two contexts, such that $\Gamma \subseteq \Gamma'$ and $\Gamma \vdash t : A$ is derivable in $\mathscr{T}'$, then $\Gamma' \vdash t : A$ is derivable in $\mathscr{T}'$.*

Proof. By induction on the derivation of $\Gamma \vdash t : A$ in $\mathscr{T}'$.

Lemma 3.2 (Key lemma) *If Γ is well-formed and $\Gamma \vdash t : A$ is derivable in $\mathscr{T}'$, then $\Gamma \vdash t : A$ is derivable in $\mathscr{T}$.*

Proof. By induction on the derivation of $\Gamma \vdash t : A$ in $\mathscr{T}'$.

- If the derivation ends with the rule (sort'), then $t = s_1$, $A = s_2$, and $\langle s_1, s_2 \rangle \in \mathscr{A}$. As Γ is well-formed, $\Gamma \vdash s_1 : s_2$ is derivable in $\mathscr{T}$ with the rules (sort) and (weak).
- If the derivation ends with the rule (var'), then t is a variable x, $\Gamma = \Gamma_1, x : A, \Gamma_2$, and $\Gamma \vdash A : s$ is derivable in the system $\mathscr{T}'$. As Γ is well-formed, $\Gamma_1 \vdash A : s'$ is derivable in $\mathscr{T}$. Thus, $\Gamma_1, x : A \vdash x : A$ is derivable in $\mathscr{T}$ with the rule (start). And, as Γ is well-formed, $\Gamma_1, x : A, \Gamma_2 \vdash x : A$ is derivable with the rule (weak).
- If the derivation ends with the rule (prod), then $t = (x : C) \to D$, $A = s_3$, $\Gamma \vdash C : s_1$ is derivable in $\mathscr{T}'$, $\Gamma, x : C \vdash D : s_2$ is derivable in $\mathscr{T}'$, and $\langle s_1, s_2, s_3 \rangle \in \mathscr{R}$. Then, as Γ is well-formed, by induction hypothesis, $\Gamma \vdash C : s_1$ is derivable in $\mathscr{T}$. Thus, $\Gamma, x : C$ is well-formed and, by induction hypothesis again, $\Gamma, x : C \vdash D : s_2$ is derivable in $\mathscr{T}$. So, $\Gamma \vdash (x : C) \to D : s_3$ is derivable in $\mathscr{T}$ with the rule (prod).
- If the derivation ends with the rule (abs), then $t = \lambda x : C\ u$, $A = (x : C) \to D$, $\Gamma \vdash C : s_1$ is derivable in $\mathscr{T}'$, $\Gamma, x : C \vdash D : s_2$ is derivable in $\mathscr{T}'$, $\Gamma, x : C \vdash u : D$ is derivable in $\mathscr{T}'$, and $\langle s_1, s_2, s_3 \rangle \in \mathscr{R}$. By induction hypothesis, $\Gamma \vdash C : s_1$ is derivable in $\mathscr{T}$. Thus, $\Gamma, x : C$ is well-formed and, by induction hypothesis again, $\Gamma, x : C \vdash D : s_2$ is derivable in $\mathscr{T}$ and $\Gamma, x : C \vdash u : D$ is derivable in $\mathscr{T}$. So, $\Gamma \vdash \lambda x : C\ u : (x : C) \to D$ is derivable in $\mathscr{T}$ with the rule (abs).

- If the derivation ends with the rule (app), then $t = u\ v$, $A = (v/x)D$, $\Gamma \vdash u : (x : C) \to D$ is derivable in $\mathscr{T}'$ and $\Gamma \vdash v : C$ is derivable in $\mathscr{T}'$. By induction hypothesis $\Gamma \vdash u : (x : C) \to D$ is derivable in $\mathscr{T}$ and $\Gamma \vdash v : C$ is derivable in $\mathscr{T}$. Hence $\Gamma \vdash u\ v : (v/x)D$ is derivable in $\mathscr{T}$, with the rule (app).
- If the derivation ends with the rule (conv), then $\Gamma \vdash t : C$ is derivable in $\mathscr{T}'$, $\Gamma \vdash A : s$ is derivable in $\mathscr{T}'$, and $C \equiv A$. By induction hypothesis, $\Gamma \vdash t : C$ is derivable in $\mathscr{T}$ and $\Gamma \vdash A : s$ is derivable in $\mathscr{T}$. Thus, $\Gamma \vdash t : A$ is derivable in $\mathscr{T}$, with the rule (conv).

Lemma 3.3 (Reordering) *Let Γ be a context, x a variable that does not occur in Γ, and Γ' a well-formed context, such that $\Gamma' \subseteq (\Gamma, x : C)$ and $\Gamma' \vdash t : A$ is derivable in $\mathscr{T}'$. Then, there exists a well-formed context Γ'', such that $\Gamma'', x : C \vdash t : A$ is derivable in $\mathscr{T}'$.*

Proof. If $x : C$ is in Γ', then we have $\Gamma' = \Gamma'_1, x : C, \Gamma'_2$, and as $\Gamma'_2 \subseteq \Gamma$, x does not occur in Γ'_2. We take $\Gamma'' = \Gamma'_1, \Gamma'_2$. By Lemma 2.4, Γ'' is well-formed and, by Lemma 3.1, $\Gamma'', x : C \vdash t : A$ is derivable in $\mathscr{T}'$.

Otherwise, we take $\Gamma'' = \Gamma'$. This context is well-formed and, by Lemma 3.1, $\Gamma'', x : C \vdash t : A$ is derivable in $\mathscr{T}'$.

Lemma 3.4 (Context curation) *If $\Gamma \vdash t : A$ is derivable in $\mathscr{T}'$, then there exists a well-formed context Δ, such that $\Delta \subseteq \Gamma$ and $\Delta \vdash t : A$ is derivable in $\mathscr{T}'$.*

Proof. By induction on the derivation of $\Gamma \vdash t : A$.

- If the derivation ends with the rule (sort'), then $t = s_1$ and $A = s_2$, such that $\langle s_1, s_2 \rangle \in \mathscr{A}$. We take the empty context for Δ, $\Delta \subseteq \Gamma$, Δ is well-formed, and $\Delta \vdash s_1 : s_2$ is derivable in $\mathscr{T}'$, with the rule (sort').
- If the derivation ends with the rule (var'), then t is a variable x, $x : A$ is an element of Γ and $\Gamma \vdash A : s$ is derivable in $\mathscr{T}'$. By induction hypothesis, there exists a well-formed context Δ_1, such that $\Delta_1 \subseteq \Gamma$ and $\Delta_1 \vdash A : s$ is derivable in $\mathscr{T}'$.

  If $x : A$ is an element of Δ_1, we take $\Delta = \Delta_1$. We have $\Delta \subseteq \Gamma$ and Δ is well-formed. Moreover $\Delta \vdash A : s$ is derivable in $\mathscr{T}'$ and Δ contains $x : A$, thus $\Delta \vdash x : A$ is derivable in $\mathscr{T}'$, with the rule (var').

  Otherwise, as $\Delta_1 \subseteq \Gamma$, Δ_1 contains no declaration of x, we take $\Delta = \Delta_1, x : A$. We have $\Delta \subseteq \Gamma$. By Lemma 3.2, $\Delta_1 \vdash A : s$ is derivable in $\mathscr{T}$, thus Δ is well-formed. Moreover, by Lemma 3.1, the judgement $\Delta \vdash A : s$ is derivable in $\mathscr{T}'$ and, as Δ contains $x : A$, $\Delta \vdash x : A$ is derivable in $\mathscr{T}'$, with the rule (var').
- If the derivation ends with the rule (prod) then $t = (x : C) \to D$, $A = s_3$, the contexts $\Gamma \vdash C : s_1$ and $\Gamma, x : C \vdash D : s_2$ are derivable in $\mathscr{T}'$, and $\langle s_1, s_2, s_3 \rangle \in \mathscr{R}$. Modulo α-equivalence, we can assume that x does not occur in Γ. By induction hypothesis, there exist two well-formed contexts Γ_1 and Γ_2, such that $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq (\Gamma, x : C)$, and the judgements $\Gamma_1 \vdash C : s_1$ and $\Gamma_2 \vdash D : s_2$ are derivable in $\mathscr{T}'$.

  By Lemma 3.3, there exists a well-formed context Γ'_2 such that $\Gamma'_2, x : C \vdash D : s_2$ is derivable in $\mathscr{T}'$. As Γ_1 and Γ'_2 contain no declaration of x, by Lemma 2.5, there exists a well-formed context Δ, such that $\Gamma_1 \subseteq \Delta$, $\Gamma'_2 \subseteq \Delta$, and Δ contains no declaration of x. We have $\Gamma_1 \subseteq \Delta$ and $\Gamma'_2, x : C \subseteq \Delta, x : C$. Thus, by Lemma 3.1, $\Delta \vdash C : s_1$ and $\Delta, x : C \vdash D : s_2$ are derivable in $\mathscr{T}'$. Thus, $\Delta \vdash (x : C) \to D : s_3$ is derivable in $\mathscr{T}'$, with the rule (prod).

- If the derivation ends with the rule (abs), then $t = \lambda x : C\ u$, $A = (x : C) \to D$, the judgements $\Gamma \vdash C : s_1$, $\Gamma, x : C \vdash D : s_2$, and $\Gamma, x : C \vdash u : D$ are derivable in $\mathscr{T}'$, and $\langle s_1, s_2, s_3 \rangle \in \mathscr{R}$. Modulo α-equivalence, we can assume that x does not occur in Γ. By induction hypothesis, there exist three well-formed contexts Γ_1, Γ_2, and Γ_3, such that $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq (\Gamma, x : C)$, $\Gamma_3 \subseteq (\Gamma, x : C)$, and the judgements $\Gamma_1 \vdash C : s_1$, $\Gamma_2 \vdash D : s_2$, and $\Gamma_3 \vdash u : D$ are derivable in $\mathscr{T}'$.

  By Lemma 3.3, there exists well-formed contexts Γ'_2 and Γ'_3, such that the judgements $\Gamma'_2, x : C \vdash D : s_2$ and $\Gamma'_3, x : C \vdash u : D$ are derivable in $\mathscr{T}'$. As Γ_1, Γ'_2, and Γ'_3 contain no declaration of x, using Lemma 2.5 twice, there exists a well-formed context Δ, such that $\Gamma_1 \subseteq \Delta$, $\Gamma'_2 \subseteq \Delta$, $\Gamma'_3 \subseteq \Delta$, and Δ contains no declaration of x. We have $\Gamma_1 \subseteq \Delta$, $\Gamma'_2, x : C \subseteq \Delta, x : C$, and $\Gamma'_3, x : C \subseteq \Delta, x : C$. Thus, by Lemma 3.1, the judgements $\Delta \vdash C : s_1$, $\Delta, x : C \vdash D : s_2$, and $\Delta, x : C \vdash u : D$ are derivable in $\mathscr{T}'$. Thus, $\Delta \vdash \lambda x : C\ u : (x : C) \to D$ is derivable in $\mathscr{T}'$, with the rule (abs).
- If the derivation ends with the rule (app) then $t = u\ v$, $A = (v/x)D$, and the judgements $\Gamma \vdash u : (x : C) \to D$ and $\Gamma \vdash v : C$ are derivable in $\mathscr{T}'$. By induction hypothesis, there exists two well-formed contexts Γ_1 and Γ_2, such that $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, and the judgements $\Gamma_1 \vdash u : (x : C) \to D$ and $\Gamma_2 \vdash v : C$ are derivable in $\mathscr{T}'$.

  By Lemma 2.5, there exists a well-formed context Δ, such that $\Gamma_1 \subseteq \Delta$, and $\Gamma_2 \subseteq \Delta$. By Lemma 3.1, the judgements $\Delta \vdash u : (x : C) \to D$ and $\Delta \vdash v : C$ are derivable in $\mathscr{T}'$. Thus, $\Delta \vdash (u\ v) : (v/x)D$ is derivable in $\mathscr{T}'$, with the rule (app).
- If the derivation ends with the rule (conv) then the judgements $\Gamma \vdash t : C$ and $\Gamma \vdash A : s$ are derivable in $\mathscr{T}'$, and $C \equiv A$. By induction hypothesis, there exists two well-formed contexts Γ_1 and Γ_2, such that $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, and the judgements $\Gamma_1 \vdash t : C$ and $\Gamma_2 \vdash A : s$ are derivable in $\mathscr{T}'$.

  By Lemma 2.5, there exists a well-formed context Δ, such that $\Gamma_1 \subseteq \Delta$ and $\Gamma_2 \subseteq \Delta$. By Lemma 3.1, the judgements $\Delta \vdash t : C$ and $\Delta \vdash A : s$ are derivable in $\mathscr{T}'$. Thus, $\Delta \vdash t : A$ is derivable in $\mathscr{T}'$, with the rule (conv).

Theorem 3.1 *If $\Gamma \vdash t : A$ is derivable in $\mathscr{T}'$, then there exists Δ, such that $\Delta \subseteq \Gamma$ and $\Delta \vdash t : A$ is derivable in $\mathscr{T}$.*

Proof. By Lemma 3.4, there exists a well-formed context Δ, such that $\Delta \subseteq \Gamma$ and $\Delta \vdash t : A$ is derivable in $\mathscr{T}'$. By Lemma 3.2, $\Delta \vdash t : A$ is derivable in $\mathscr{T}$.

Example. Consider two sorts $*$ and $\Box$ and an axiom $* : \Box$. From the derivation of the judgement, $z : nat, nat : * \vdash z : nat$, we extract the context $nat : *, z : nat$.

And from the derivation of $nat : *, x : (* *), z : nat \vdash z : nat$, we also extract the context $nat : *, z : nat$.

Acknowledgements

The author wants to thank Frédéric Blanqui, Herman Geuvers, and Claudio Sacerdoti Coen for useful and lively discussions about the various presentations of type theory.

References

[1] H. Barendregt (1992): *Lambda calculi with types*. In S. Abramsky, D.M. Gabbay & T.S.E. Maibaum, editors: *Handbook of Logic in Computer Science*, 2, Oxford University Press, pp. 117–309.

[2] F. Blanqui (2001): *Definitions by Rewriting in the Calculus of Constructions*. In: *Logic in Computer Science*, IEEE Computer Society, pp. 9–18, doi:10.1109/LICS.2001.932478.

[3] C. Sacerdoti Coen (2004): *Mathematical Libraries as Proof Assistant Environments*. In A. Asperti, G. Bancerek & A. Trybulec, editors: *Mathematical Knowledge Management*, *Lecture Notes in Computer Science* 3119, Springer, pp. 332–346, doi:10.1007/978-3-540-27818-4_24.

[4] T. Coquand & G. Huet (1988): *The Calculus of Constructions*. *Information and Computation* 76(2/3), pp. 95–120, doi:10.1016/0890-5401(88)90005-3.

[5] D. Cousineau & G. Dowek (2007): *Embedding Pure Type Systems in the Lambda-Pi-Calculus Modulo*. In S. Ronchi Della Rocca, editor: *Typed Lambda Calculi and Applications*, *Lecture Notes in Computer Science* 4583, Springer, pp. 102–117, doi:10.1007/978-3-540-73228-0_9.

[6] H. Geuvers, R. Krebbers, J. McKinna & F. Wiedijk (2010): *Pure Type Systems without Explicit Contexts*. In K. Crary & M. Miculan, editors: *Logical Frameworks and Meta-languages: Theory and Practice*, *Electronic Proceedings in Theoretical Computer Science* 34, Open Publishing Association, pp. 53–67, doi:10.4204/EPTCS.34.6.

[7] H. Geuvers & M.-J. Nederhof (1991): *Modular proof of strong normalization for the calculus of constructions*. *Journal of Functional Programming* 1(2), pp. 155–189, doi:10.1017/S0956796800020037.