SUBJECT REDUCTION FOR PURE TYPE SYSTEMS

Charlotte Marchal | 261031516 Dashiell Rich | 261002837
Milton Rosenbaum | 260972050 Zhaoshen Zhai | 261003108

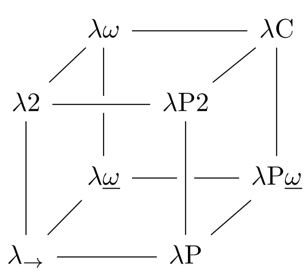
Abstract. Following [GN91] and [Bar91], we study the basics of *pure type systems*, which abstract many of the constructs found in the eight systems of the λ-*cube*. We start with a brief introduction to the systems of the λ-cube, discuss their expressive power, and introduce pure type systems as a unifying framework in which they can be studied. We then give a detailed proof of subject reduction for arbitrary pure type systems.

Introduction. Subject reduction is a crucial property of a type system that guarantees its 'computational consistency' by ensuring that reductions of a well-typed expression remains well-typed, and also supports the slogan that 'well-typed programs do not go wrong'. It is thus desirable that it can be stated and proven uniformly across many different type systems, and this is the goal of the present note.

To this end, we follow [Bar91] and work within the framework of *pure type systems*, which is an abstraction of type systems based on the idea of 'dependencies' between terms and types. In the simply-typed λ-calculus $\lambda_\to$, terms depend on terms: $F\,x$ is a term depending on the term x, and we can abstract this dependency to a function $\lambda x{:}\alpha_1.(F\,x)$ of type $\alpha_1 \to \alpha_2$, where $F\,x{:}\alpha_2$. The other three ways that terms and types can be mutually dependent are present in other type systems, which all extend $\lambda_\to$.

- In the polymorphic λ-calculus $\lambda 2$, the term $I_\alpha \coloneqq \lambda x{:}\alpha.x$ depends on the type α, and this abstracts to $\lambda\alpha{:}{*}.I_\alpha$ of type $\forall\alpha{:}{*}.(\alpha\to\alpha)$. Here, '$\alpha{:}{*}$' formalizes '$\alpha$ is a type' within $\lambda 2$.
- In the (weak) higher-order λ-calculus $\lambda\underline{\omega}$, the type $\alpha\to\alpha$ depends on the type α, and this abstracts to a *constructor* $\lambda\alpha{:}{*}.(\alpha\to\alpha)$ of *kind* $*\to*$. Similarly, in the λ-calculus λP of dependent types, the type $\alpha_1^n \to \alpha_2$ depends on the term n, and abstracts to a constructor $\lambda n{:}\mathbb{N}.(\alpha_1^n\to\alpha_2)$ of kind $\mathbb{N}\to *$.

Assuming that the type systems that we care about all extend $\lambda_\to$, and hence terms can depend on terms, we obtain a total of $8 = 2^3$ type systems with all possible combinations of dependencies, called the λ-*cube*:



Type system	Dependencies			
$\lambda_\to$	$(*,*)$			
$\lambda 2$	$(*,*)$	$(\square,*)$		
$\lambda\underline{\omega}$	$(*,*)$		$(\square,\square)$	
$\lambda\omega = \lambda 2\underline{\omega}$	$(*,*)$	$(\square,*)$	$(\square,\square)$	
λP	$(*,*)$			$(*,\square)$
λP2	$(*,*)$	$(\square,*)$		$(*,\square)$
λP$\underline{\omega}$	$(*,*)$		$(\square,\square)$	$(*,\square)$
λC $= \lambda$P2$\underline{\omega}$	$(*,*)$	$(\square,*)$	$(\square,\square)$	$(*,\square)$

The systems λP$\underline{\omega}$, λP2, and $\lambda\omega \coloneqq \lambda 2\underline{\omega}$ have three kinds of dependencies, while the strongest system of them all, the *calculus of constructions* λC $\coloneqq \lambda$P2$\underline{\omega}$, have all four kinds of dependencies.

To explain the table on the right, we observe that due to the mutual dependencies of terms and types, it is no longer natural to separate their definitions. Instead, we propose a uniform object, called a *pseudoterm*, and consider a judgement $M{:}N$ between pseudoterms M and N. These pseudoterms include terms, types, and kinds, so this begs the question: what is $*$? If $*{:}*$, then one might encounter a Russell-like paradox of the 'type of all types', so instead, we introduce a new symbol $\square$, the 'sort of all kinds', and assert that $*{:}\square$. Similarly, $(*\to*){:}\square$ and $(\mathbb{N}\to*){:}\square$, so sorts also capture the identity of (higher order) type constructors.

In the table, the notation (s_1,s_2) means that inhabitants of s_2 can depend on those of s_1, and moreover, that we can abstract over inhabitants of s_1 and output those of s_2. For instance, the '$(\square,*)$' in $\lambda 2$ indicate that terms (inhabitants of $*$) depend on types (inhabitants of $\square$), and that we can abstract over types and output terms (say, in $\lambda\alpha{:}{*}.I_\alpha$). Note that $(*\to*){:}\square$, so this allows for higher-order polymorphism as well.

Date: April 22, 2025.
Extended abstracted for the final project for Comp527: Logic and Computation taught by Professor Brigitte Pientka.
Link to presentation: **TODO**

Pure type systems. To delineate the hierarchy between terms, types, and kinds, one is naturally inclined to start abstractly and axiomatize the notion of a type system.

Definition 1. A *pure type system* is a tuple $\sigma \coloneqq (\mathcal{C}, \mathcal{V}, \mathcal{S}, \mathcal{A}, \mathcal{R})$ consisting of a set $\mathcal{C}$ of *constants*, a set $\mathcal{V}$ of *variables*, a set $\mathcal{S} \subseteq \mathcal{C}$ of *sorts*, a set $\mathcal{A} \subseteq \mathcal{C}^2$ of *axioms*, and a set $\mathcal{R} \subseteq \mathcal{S}^3$ of *rules*.

Intuitively, *sorts* are universes imposing some sort (pun intended) of classification, *constants* are symbols living in a sort/constant as dictated by *axioms* (for instance, $0{:}\mathbb{N}$, $\mathbb{N}{:}*$, and $*{:}\Box$), and the *rules* restrict the formation of abstractions as motivated above. For the systems in the λ-cube, we write $(s_1, s_2) \coloneqq (s_1, s_2, s_2)$. See [Bar91] or [Bar92] for how the eight systems in the λ-cube can be interpreted as pure type systems.

Let us now proceed by defining the necessary notions to state and prove subject reduction for an arbitrary pure type system σ, for which we will need two lemmas in [GN91]. We also invite the reader to see how the following restrict to the standard notions/proofs for systems in the λ-cube.

Definition 2. The collection of σ-*pseudoterms* is defined by $T \coloneqq \mathcal{V} \,|\, \mathcal{C} \,|\, (T\,T) \,|\, (\lambda\mathcal{V}{:}T.T) \,|\, (\Pi\mathcal{V}{:}T.T)$. Pairs $(A, B) \in T^2$ are called σ-*assignments*, written $A{:}B$, and a finite sequence thereof is called a σ-*pseudocontext*.

Definition 3. The β-*reduction* relation is the least relation $\to_\beta$ on σ-terms satisfying the following rules for all σ-terms A, A', A'': $(\lambda x{:}A.A')A'' \to_\beta A'[A''/x]$, and the *congruence rules* $A\,A' \to_\beta A\,A''$, $A'\,A \to_\beta A''\,A$, $\lambda x{:}A.A' \to_\beta \lambda x{:}A.A''$, $\lambda x{:}A'.A \to_\beta \lambda x{:}A''.A$, $\Pi x{:}A.A' \to_\beta \Pi x{:}A.A''$, and $\Pi x{:}A'.A \to_\beta \Pi x{:}A''.A$.

Notation 4. We write $\twoheadrightarrow_\beta$ for the reflexive and transitive closure of $\to_\beta$, and $=_\beta$ for the equivalence relation generated by $\twoheadrightarrow_\beta$. A σ-term of the form $(\lambda x{:}A.A')A''$ is called a β-*redex*. If $\Gamma \coloneqq (x_1{:}A_1, \ldots, x_n{:}A_n)$ and $\Gamma' \coloneqq (x_1{:}A'_1, \ldots, x_n{:}A'_n)$ are σ-pseudocontexts, we also write $\Gamma \to_\beta \Gamma$ if $A_i \to_\beta A'_i$ for each $i \le n$.

Definition 5. Let Γ be a σ-pseudocontext and let M, N be σ-pseudoterms. We say that Γ *proves* $M{:}N$, and write $\Gamma \vdash M{:}N$, if there is a finite well-founded tree $\mathcal{D}$, called a *derivation*, such that the following hold.

1. Vertices of $\mathcal{D}$ are of the form $\Delta \vdash A{:}B$, where A and B are σ-pseudoterms and Δ is a σ-pseudocontext.
2. The root of $\mathcal{D}$ is $\Gamma \vdash M{:}N$ and the leaves of $\mathcal{D}$ are instances of $\vdash c{:}c'$, where $(c, c') \in \mathcal{A}$.
3. Each interior vertex of $\mathcal{D}$ is a conclusion of an *inference rule*, whose successors are exactly the premises.

The inference rules of σ are as follows. Below, $s \in \mathcal{S}$, $x \in \mathcal{V} \setminus \operatorname{dom}\Gamma$, $(s_1, s_2, s_3) \in \mathcal{R}$, and $C =_\beta C'$.

$$\frac{\Gamma \vdash A{:}s}{\Gamma, x{:}A \vdash x{:}A}\ \textsc{Init} \qquad \frac{\Gamma \vdash A{:}s \quad \Gamma \vdash B{:}C}{\Gamma, x{:}A \vdash B{:}C}\ \textsc{Weak} \qquad \frac{\Gamma \vdash B{:}C \quad \Gamma \vdash C'{:}s}{\Gamma \vdash B{:}C'}\ \textsc{Conv} \qquad \frac{\Gamma \vdash B_1{:}s_1 \quad \Gamma, x{:}B_1 \vdash B_2{:}s_2}{\Gamma \vdash (\Pi x{:}B_1.B_2){:}s_3}\ \Pi\text{-RULE}$$

$$\frac{\Gamma \vdash B_1{:}s_1 \quad \Gamma, x{:}B_1 \vdash B_2{:}s_2 \quad \Gamma, x{:}B_1 \vdash C{:}B_2}{\Gamma \vdash (\lambda x{:}B_1.C){:}(\Pi x{:}B_1.B_2)}\ \lambda\text{-RULE} \qquad \frac{\Gamma \vdash B_1{:}(\Pi x{:}C_1.C_2) \quad \Gamma \vdash B_2{:}C_1}{\Gamma \vdash B_1\,B_2{:}C_2[B_2/x]}\ \textsc{App}$$

Definition 6. If $\Gamma \vdash M{:}N$, then Γ is called a σ-*context* and M, N are called σ-*terms*.

Lemma 7 (Substitution Lemma; [GN91, Lemma 17])**.** *Let Γ and $\Gamma_1, y{:}A, \Gamma_2$ be σ-contexts and let A, M, N, P be σ-terms. If $\Gamma_1, y{:}A, \Gamma_2 \vdash M{:}N$ and $\Gamma \vdash P{:}A$, then $(\Gamma_1, \Gamma_2)[P/y] \vdash M[P/y]{:}N[P/y]$.*

Lemma 8 (Stripping Lemma; [GN91, Lemma 19])**.** *Let Γ be a σ-context and let M, N, P be σ-terms.*

1. *If $\Gamma \vdash c{:}P$ where $c \in \mathcal{C}$, then $P =_\beta c'$ and $(c, c') \in \mathcal{A}$ for some $c' \in \mathcal{C}$.*
2. *If $\Gamma \vdash x{:}P$ where $x \in \mathcal{V}$, then $P =_\beta Q$ for some σ-term Q such that $(x{:}Q) \in \Gamma$.*
3. *If $\Gamma \vdash (\Pi x{:}M.N){:}P$, then $\Gamma \vdash M{:}s_1$, $\Gamma, x{:}M \vdash N{:}s_2$, and $P =_\beta s_3$ for some $(s_1, s_2, s_3) \in \mathcal{R}$.*
4. *If $\Gamma \vdash (\lambda x{:}M.N){:}P$, then $\Gamma \vdash M{:}s_1$, $\Gamma, x{:}M \vdash Q{:}s_2$, $\Gamma, x{:}M \vdash N{:}Q$, $\Gamma \vdash P{:}s_3$, and $P =_\beta \Pi x{:}M.Q$ for some $(s_1, s_2, s_3) \in \mathcal{R}$ and σ-term Q.*
5. *If $\Gamma \vdash M\,N{:}P$, then $\Gamma \vdash M{:}(\Pi x{:}C_1.C_2)$, $\Gamma \vdash N{:}C_1$, and $P =_\beta C_2[N/x]$ for some σ-terms C_1 and C_2.*

Theorem 9 (Subject Reduction; [GN91, Lemma 22])**.** *Let Γ, Γ' be σ-contexts and let M, M', N be σ-terms.*

1. *If $\Gamma \vdash M{:}N$ and $M \twoheadrightarrow_\beta M'$, then $\Gamma \vdash M'{:}N$.*
2. *If $\Gamma \vdash M{:}N$ and $\Gamma \twoheadrightarrow_\beta \Gamma'$, then $\Gamma' \vdash M{:}N$.*

Let us close by mentioning that beyond providing uniform proofs for many statements in the λ-cube (see [GN91] for a modular proof of strong normalization for systems in the λ-cube), pure type systems remain an active area of research nowadays. See [Ter95], [BHS01], and [BG05] for surveys and extensions.

Appendix. We sketch a proof of the Stripping Lemma, since it is crucial to the proof of subject reduction, and use it to provide the full proof of subject reduction for an arbitrary pure type system σ.

Proof of Lemma 8. Let $\mathcal{D}$ denote the derivation $\Gamma \vdash A : P$ in each of the five cases. Tracing $\mathcal{D}$ upwards from the root, taking the left branch for App and Conv, and the right branch for Weak, Π-rule, and λ-rule, we alternate between instances of Conv and Weak before the σ-term A is introduced in either an axiom, Init, Π-rule, λ-rule, or App; these five cases correspond respectively with the five cases for $\mathcal{D}$.

Due to possible applications of Weak and Conv, the vertex of $\mathcal{D}$ right after A is introduced has the form $\Gamma' \vdash A : P'$ for some initial segment Γ' of Γ and some σ-term P' such that $P' =_\beta P$. Then, inspecting the (unique) rule introducing A, we can conclude the premises of said rule and the desired form of P'. ■

Proof of Theorem 9. We proceed by simultaneous induction on the derivation $\mathcal{D} : \Gamma \vdash M : N$ when $M \to_\beta M'$ and $\Gamma \to_\beta \Gamma'$; the general case follows by iteration. We first prove (1), and split into cases with similar proofs.

- If $\mathcal{D}$ ends with Init, then there is no redex in M. If $\mathcal{D}$ ends with Conv, then there are derivations $\mathcal{D}_1 : \Gamma \vdash M : N'$ and $\mathcal{D}_2 : \Gamma \vdash N' : s$ for some $s \in \mathcal{S}$ and some σ-term N' such that $N' =_\beta N$. By IH$_1$, we have $\Gamma \vdash M' : N'$, on which Conv with $\mathcal{D}_2$ gives $\Gamma \vdash M' : N$. The case when $\mathcal{D}$ ends with Weak is similar.
- If $\mathcal{D}$ ends with Π-rule, say with $M = \Pi x : B_1 . B_2$, then the Stripping Lemma furnish some $(s_1, s_2, s_3) \in \mathcal{R}$ and derivations $\mathcal{D}_1 : \Gamma \vdash B_1 : s_1$ and $\mathcal{D}_2 : \Gamma, x : B_1 \vdash B_1 : s_2$ such that $N =_\beta s_3$. By definition of $\to_\beta$, two cases occur: if there is a σ-term B_1' such that $B_1 \to_\beta B_1'$, then by IH$_1$ on $\mathcal{D}_1$, we have $\mathcal{D}_1' : \Gamma \vdash B_1' : s_1$. Moreover, IH$_2$ on $\mathcal{D}_2$ gives $\mathcal{D}_2' : \Gamma, x : B_1' \vdash B_2 : s_2$, so applying Π-rule on $\mathcal{D}_1'$ and $\mathcal{D}_2'$ gives $\Gamma \vdash (\Pi x : B_1' . B_2) : s_3$, on which Conv gives $\Gamma \vdash (\Pi x : B_1' . B_2) : N$. The second case when $B_2 \to_\beta B_2'$ for some σ-term B_2' is the same (in fact, easier). The case when $\mathcal{D}$ ends with λ-rule is similar (and again has two subcases).
- If $\mathcal{D}$ ends with App, say with $M = B_1\, B_2$, then reductions within either B_1 or B_2 are trivial. Thus, we can take $x \in \mathcal{V} \setminus \mathrm{dom}\,\Gamma$ such that $B_1 = \lambda x : A_1 . A_2$, and assume $M = (\lambda x : A_1 . A_2) B_2 \to_\beta A_2[B_2/x]$. The Stripping Lemma then furnish σ-terms C_1 and C_2 such that $N =_\beta C_2[B_2/x]$ and derivations $\mathcal{D}_1 : \Gamma \vdash (\lambda x : A_1 . A_2) : (\Pi x : C_1 . C_2)$ and $\mathcal{D}_2 : \Gamma \vdash B_2 : C_1$. Again, the Stripping Lemma applied to $\mathcal{D}_1$ then furnish $(s_1, s_2, s_3) \in \mathcal{R}$, a σ-term C_2' such that $\Pi x : C_1 . C_2 =_\beta \Pi x : A_1 . C_2'$, and derivations $\mathcal{E}_1 : \Gamma \vdash A_1 : s_1$, $\mathcal{E}_2 : \Gamma, x : A_1 \vdash C_2' : s_2$, and $\mathcal{E}_3 : \Gamma, x : A_1 \vdash A_2 : C_2'$. Observe that $A_1 =_\beta C_1$, so Conv on $\mathcal{D}_2$ and $\mathcal{E}_1$ gives $\mathcal{D}_0 : \Gamma \vdash B_2 : A_1$, and using the Substitution Lemma with $(\mathcal{D}_0, \mathcal{E}_2)$ and $(\mathcal{D}_0, \mathcal{E}_3)$ give $\mathcal{E}_2' : \Gamma \vdash C_2'[B_2/x] : s_2$ and $\mathcal{E}_3' : \Gamma \vdash A_2[B_2/x] : C_2'[B_2/x]$; note that $\Gamma[B_2/x] = \Gamma$ since $x \notin \mathrm{dom}\,\Gamma$. Finally, since $C_2 =_\beta C_2'$ and $N =_\beta C_2[B_2/x]$, applying Conv on $\mathcal{E}_2'$ and $\mathcal{E}_3'$ gives $\Gamma \vdash A_2[B_2/x] : N$.

For (2), if the last rule of $\mathcal{D}$ is either App, Conv, Π-rule, or λ-rule, then we are done by IH$_2$; indeed, Γ is unchanged for App and Conv, and in Π-rule and λ-rule, reductions take place within Γ. Suppose that the last rule of $\mathcal{D}$ is Init or Weak, so with the notation of Definition 5, $\Gamma = \Gamma_0, x : A$ for some σ-term A and $x \in \mathcal{V}$. If the reduction occurs within Γ_0, then we are done by IH$_2$. Otherwise, $A \to_\beta A'$ for some σ-term A.

- If $\mathcal{D}$ ends with Init, then $\Gamma_0 \vdash A : s$. Applying IH$_1$, we have $\Gamma_0 \vdash A' : s$, and hence $\Gamma_0, x : A' \vdash x : A'$ from Init. Since $A \to_\beta A'$, we see that $A =_\beta A'$, so $\Gamma_0, x : A' \vdash x : A$ by Conv, as desired.
- If $\mathcal{D}$ ends with Weak, then there are derivations $\mathcal{D}_1 : \Gamma_0 \vdash A : s$ and $\mathcal{D}_2 : \Gamma_0 \vdash B : C$. Applying IH$_1$, we obtain a derivation $\mathcal{D}_1' : \Gamma_0 \vdash A' : s$, and applying Weak on $\mathcal{D}_1'$ and $\mathcal{D}_2$ gives $\Gamma_0, x : A' \vdash B : C$. ■

References

[GN91] H. Geuvers and M. Nederhof, *Modular proof of strong normalization for the calculus of constructions*, Journal of Functional Programming **1** (1991), no. 2, 155-189.

[Bar91] H. Barendregt, *Introduction to Generalized Type Systems*, Journal of Functional Programming **1** (1991), no. 2, 125-154.

[Bar92] ———, *Lambda calculi with types* (S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, eds.), Handbook of Logic in Computer Science, vol. 2, Oxford University Press, 1992.

[Ter95] J. Terlouw, *Strong normalization in type systems: a model theoretical approach*, Annals of Pure and Applied Logic **73** (1995), 53-78.

[BHS01] G. Barthe, J. Hatcliff, and M. H. B. Sørensen, *An induction principle for pure type systems*, Theoretical Computer Science **266** (2001), 773-818.

[BG05] B. Barras and B. Grégoire, *On the Role of Type Decorations in the Calculus of Inductive Constructions* (L. Ong, ed.), Computer Science Logic, CSL 2005, Lecture Notes in Computer Science, vol. 3634, Springer Berlin Heidelberg, 2005.