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 which supports the slogan that 'well-typed programs do not go wrong'. It is thus desirable that we can prove it uniformly across many different type systems, and this is the goal of the present note.

To this end, **TODO:** review λ-cube and the dependencies.

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

Notation 2. Throughout, let σ be denote an arbitrary pure type system.

TODO: link this with the λ-cube by interpreting $\lambda_{\to}$ as a PTS.

Definition 3. The collection of σ-*pseudoterms* is defined by $T := \mathcal{V} \mid \mathcal{C} \mid (T\,T) \mid (\lambda \mathcal{V}{:}T.T) \mid (\Pi \mathcal{V}{:}T.T)$. Pairs $(A, B) \in T^2$ are called σ-*assignments*, written $A{:}B$, and a finite sequence thereof is called a σ-*pseudocontext*.

Definition 4. The β-*reduction* relation is the least relation on σ-terms satisfying the following for all σ-terms A, A', A'': the *principal reduction rule* $(\lambda x{:}A.A')A'' \to_\beta A'[A''/x]$, and the *congruence rules* $A\,A' \succ A\,A''$, $A'\,A \succ A''\,A$, $\lambda x{:}A.A' \succ \lambda x{:}A.A''$, $\lambda x{:}A'.A \succ \lambda x{:}A''.A$, $\Pi x{:}A.A' \succ \Pi x{:}A.A''$, and $\Pi x{:}A'.A \succ \Pi x{:}A''.A$.

Notation 5. 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*.

Definition 6. Fix a σ-pseudocontext Γ and σ-pseudoterms M and N. 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 $\Gamma \vdash M{:}N$.
3. Each interior vertex of $\mathcal{D}$ is a conclusion of an *inference rule*, whose successors are exactly the premises.
4. The leaves of $\mathcal{D}$ are instances of an *axiom* $c{:}c'$ where $(c, c') \in \mathcal{A}$.

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

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

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

Definition 7. A σ-pseudocontext Γ is a σ-*context* if there are σ-pseudoterms A, B such that $\Gamma \vdash A{:}B$. Dually, a σ-pseudoterms A is a σ-*term* if there is a σ-context Γ and a σ-pseudoterm B such that $\Gamma \vdash A{:}B$.

Date: April 20, 2025.
Extended abstracted for the final project for Comp527: Logic and Computation taught by Professor Brigitte Pientka.
Link to presentation: **TODO**

Lemma 8 (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 9 (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 $\Gamma \vdash M{:}(\Pi x{:}A.B)$, $\Gamma \vdash N{:}A$, and $P =_\beta B[N/x]$ for some σ-terms A and B.*

Theorem 10 (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 $\Gamma' \vdash M{:}N$.*

Proof. **TODO** ■

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.