SUBJECT REDUCTION FOR PURE TYPE SYSTEMS

ZHAOSHEN ZHAI

Abstract. Following [SU06], we give a detailed proof of *subject reduction* for arbitrary *pure type systems*, which abstract many of the basic constructs found in, say, the simply-typed λ-calculus ($\lambda_\to$), the polymorphic λ-calculus ($\lambda 2$), the λ-calculus with type constructors ($\lambda\underline{\omega}$), and the λ-calculus with dependent types ($\lambda\mathbf{P}$).

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, we start from the beginning[1] with the *simply-typed λ-calculus* $\lambda_\to$, in which we prove subject reduction. We then progress to more complicated type systems (in particular, $\lambda 2$, $\lambda\underline{\omega}$, and $\lambda\mathbf{P}$) to illustrate some concepts not present in $\lambda_\to$, and along the way, we also mention the *λ-cube* to provide some motivation for *pure type systems*, which abstract the constructs in all of the previous systems. Finally, we prove subject reduction for pure type systems. We will not discuss any of these systems in length, but refer the interested reader to [SU06] for general type theory and [Bar91] for actual applications of pure type systems.

1. The Simply-typed λ-calculus: $\lambda_\to$

Throughout, fix a countably infinite set V, whose element we call *(propositional/λ-)variables*.

Definition 1.1. A *simple type* is a formula in the language $\{\to\}$. We let $\Phi_\to$ denote the set of simple types.

Definition 1.2. A *λ-term* is a string defined by the grammar[2] $M \coloneqq x \mid M\,M \mid (\lambda x^\tau M)$, where $\tau \in \Phi_\to$. We denote by Λ the set of λ-terms. The set of *free variables* of a λ-term M is defined inductively by

$$FV(x) \coloneqq \{x\}, \qquad FV(\lambda x^\tau M) \coloneqq FV(M) \setminus \{x\}, \qquad FV(M\,N) \coloneqq FV(M) \cup FV(N).$$

Remark 1.3. We always consider λ-terms under α-conversion. Basically, we can freely change the bound variable x in λx without modifying the term, but see [SU06, Section 1.2] for the formal definition.

Definition 1.4. A *context* is a finite set $\Gamma \coloneqq \{x_1 : \tau_1, \dots, x_n : \tau_n\}$ of pairs $(x_i : \tau_i)$, where each $x_i \in V$ and each $\tau_i \in \Phi_\to$. If $(x : \tau) \in \Gamma$, we write $\Gamma(x) = \tau$, and we let

$$\operatorname{dom}\Gamma \coloneqq \{x \in V : \Gamma(x) = \tau \text{ for some } \tau \in \Phi_\to\} \quad \text{and} \quad \operatorname{im}\Gamma \coloneqq \{\tau \in \Phi_\to : \Gamma(x) = \tau \text{ for some } x \in V\}.$$

A *judgement* is a triple $\Gamma \vdash M : \tau$ consisting of a context Γ, a λ-term M, and a simple type τ.

Definition 1.5. We say that a judgement $\Gamma \vdash M : \tau$ is *derivable in* $\lambda_\to$ if there is a finite tree of judgements rooted at $\Gamma \vdash M : \tau$, whose leaves are instances of Var, and such that the children of each internal node is obtained from the rules Abs or App read bottom-up.

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ \textsc{Var} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x^\sigma M) : \sigma \to \tau}\ \textsc{Abs} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash (M\,N) : \tau}\ \textsc{App}$$

The rules Abs and App can only be applied when $x \notin \operatorname{dom}\Gamma$.

Lemma 1.6 (Generation Lemma for $\lambda_\to$)**.** *Suppose that*[3] $\Gamma \vdash M : \tau$.

(1) If $M = x$, then $\Gamma(x) = \tau$.

Date: April 2, 2025.

Notes for a project for Comp527: Logic and Computation taught by Professor Brigitte Pientka, with Charlotte Marchal, Dashiell Rich and Milton Rosenbaum.

[1] As Professor Pientka would say: 'We'll start slow'.

[2] Sometimes, people write $\lambda x : \tau M$ instead of $\lambda x^\tau M$, but we find the former too hard to parse.

[3] When we assert '$\Gamma \vdash M : \tau$', we mean that it is derivable in the current type system under consideration.

(2) If $M = P\,Q$, then $\Gamma \vdash P : \sigma \to \tau$ and $\Gamma \vdash Q : \sigma$ for some $\sigma \in \Phi_{\to}$.
(3) If $M = \lambda x^{\sigma} N$ and $x \notin \operatorname{dom}\Gamma$, then $\tau = \sigma \to \rho$ and $\Gamma, x : \sigma \vdash N : \rho$ for some $\rho \in \Phi_{\to}$.

Proof. Since the root of the derivation tree for $\Gamma \vdash M : \tau$ determines the shape of M, we see that (1) follows from Var and (2) follows from App. For (3), the child of the root must be obtained from Abs and is of the form $\Gamma, y : \sigma \vdash N' : \rho$, where $\lambda x^{\sigma} N = \lambda y^{\sigma} N'$, so clearly $\tau = \sigma \to \rho$. Moreover, note that $N' = N[y/x]$, so $\Gamma, y : \sigma \vdash N[y/x] : \rho$, and finally substituting x for y back gives $\Gamma, x : \sigma \vdash N : \rho$, as desired. ■

Lemma 1.7 (Change of Context)**.** *If $\Gamma \vdash M : \tau$ and $\Gamma(x) = \Gamma'(x)$ for all $x \in FV(M)$, then $\Gamma' \vdash M : \tau$.*

Proof. By induction on M. If $M = x$, then $\Gamma'(x) = \Gamma(x) = \tau$ by Lemma 1.6.1, and hence $\Gamma' \vdash x : \tau$ by Var. If $M = P\,Q$, then by Lemma 1.6.2, we have $\Gamma \vdash P : \sigma \to \tau$ and $\Gamma \vdash Q : \sigma$ for some $\sigma \in \Phi_{\to}$. By induction, we see that $\Gamma' \vdash P : \sigma \to \tau$ and $\Gamma' \vdash Q : \sigma$, on which App gives $\Gamma' \vdash M : \tau$. Lastly, if $M = \lambda x^{\sigma} N$, we can choose $x \notin \operatorname{dom}\Gamma \cup \operatorname{dom}\Gamma'$, so that $\tau = \sigma \to \rho$ and $\Gamma, x : \sigma \vdash N : \rho$ by Lemma 1.6.3. By induction, we see that $\Gamma', x : \sigma \vdash N : \rho$, on which Abs gives the desired as $\Gamma' \vdash M : \tau$. ■

We can think of the Change of Context lemma as a generalizing weakening as we can take $\Gamma' \coloneqq \Gamma, y : \sigma$ for $y \notin FV(M)$, and this is exactly how we use it below.

Lemma 1.8 (Substitution Lemma for $\lambda_{\to}$)**.** *If $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma$, then $\Gamma \vdash M[N/x] : \tau$.*

Proof. By induction on M. If $M = y$ and $x \neq y$, then $\Gamma(y) = \tau$ and $M[N/x] = y$, so that $\Gamma \vdash y : \tau$ by Var. If $x = y$, then $\Gamma(x) = \sigma$ and $M[N/x] = N$, so $\tau = \sigma$ and $\Gamma \vdash N : \sigma$ by assumption. If $M = P\,Q$, then by Lemma 1.6.2, we have $\Gamma, x : \sigma \vdash P : \rho \to \tau$ and $\Gamma, x : \sigma \vdash Q : \rho$ for some $\rho \in \Phi_{\to}$. By induction, we see that $\Gamma \vdash P[N/x] : \rho \to \tau$ and $\Gamma \vdash Q[N/x] : \rho$, on which App gives $\Gamma \vdash M[N/x] : \tau$.

Lastly, if $M = \lambda y^{\eta} M'$ where $y \notin \operatorname{dom}\Gamma \cup \{x\} \cup FV(N)$, then by Lemma 1.6.3, there is some $\rho \in \Phi_{\to}$ such that $\tau = \eta \to \rho$ and $\Gamma, x : \sigma, y : \eta \vdash M' : \rho$. By Lemma 1.7, we can weaken $\Gamma \vdash N : \sigma$ to $\Gamma, y : \eta \vdash N : \sigma$, so by induction[4] we have $\Gamma, y : \eta \vdash M'[N/x] : \rho$, and we can apply Abs to get $\Gamma \vdash M[N/x] : \tau$. ■

Definition 1.9. A relation $\succ$ on Λ is *compatible* if for any $M, N \in \Lambda$ with $M \succ N$, we have $MP \succ NP$ and $PM \succ PN$ for each $P \in \Lambda$, and $\lambda x^{\tau} M \succ \lambda x^{\tau} N$ for each $x \in V$ and $\tau \in \Phi_{\to}$.

Definition 1.10. The least compatible relation $\to_{\beta}$ on Λ such that $(\lambda x^{\tau} M)N \to_{\beta} M[N/x]$ for all $M, N \in \Lambda$ is called *β-reduction*. We say that $(\lambda x^{\tau} M)N$ is a *β-redex* and that $M[N/x]$ arises by *contracting* the redex.

Notation 1.11. For any relation $\to_{\bullet}$ on a set X, we let $\twoheadrightarrow_{\bullet}^{+}$ denote the transitive closure, let $\twoheadrightarrow_{\bullet}$ denote the transitive and reflexive closure, and let $=_{\bullet}$ denote the least equivalence relation containing $\twoheadrightarrow_{\bullet}$.

Theorem 1.12 (Subject Reduction for $\lambda_{\to}$)**.** *If $\Gamma \vdash M : \tau$ and $M \twoheadrightarrow_{\beta} N$, then $\Gamma \vdash N : \tau$.*

Proof. If $M = (\lambda x^{\sigma} P)Q$ and $N = P[Q/x]$ for some $x \notin \operatorname{dom}\Gamma$, then we have $\Gamma, x : \sigma \vdash P : \tau$ and $\Gamma \vdash Q : \sigma$ by Lemma 1.6.2 and 1.6.3, so $\Gamma \vdash N : \tau$ by Lemma 1.8. The general case follows by induction on $\twoheadrightarrow_{\beta}$. ■

2. The polymorphic λ-calculus: $\lambda 2$

Definition 2.1.

Lemma 2.2.

Theorem 2.3 (Subject Reduction for $\lambda 2$)**.**

3. The λ-calculus with type constructors: $\lambda\underline{\omega}$

Definition 3.1.

Lemma 3.2.

Theorem 3.3 (Subject Reduction for $\lambda\underline{\omega}$)**.**

[4]Note that our contexts are unordered, so we have exchange implicitly.

4. The λ-calculus with Dependent Types: $\lambda\mathbf{P}$

Definition 4.1.

Lemma 4.2.

Theorem 4.3 (Subject Reduction for $\lambda\mathbf{P}$)**.**

5. The λ-cube and beyond: Pure Type Systems

Definition 5.1.

Lemma 5.2.

Theorem 5.3 (Subject Reduction for Pure Type Systems)**.**

References

[SU06] M. H. Sørensen and P. Urzyczyin, *Lectures on the Curry-Howard Isomorphism*, Studies in Logic and the Foundations of Mathematics, Elsevier, 2006.

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

Department of Mathematics and Statistics, McGill University, 805 Sherbrooke Street West, Montreal, QC, H3A 0B9, Canada

Email address: zhaoshen.zhai@mail.mcgill.ca