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 and generalize many of the basic constructs found in, say, the simply-typed λ-calculus, the polymorphic λ-calculus (System $\mathbf{F}$), the λ-calculus with dependent types $\lambda\mathbf{P}$, and much more.

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, System $\mathbf{F}$ and $\lambda\mathbf{P}$) to illustrate some more subtleties and concepts not present in $\lambda_\to$, and finally define *pure type systems* and prove subject reduction therein. We will not discuss any of these systems in length (or, at all...), but refer the interested reader to [SU06] for general type theory and [Bar91] for the motivation and applications of pure type systems.

1. The Simply-typed λ-calculus

Throughout, fix a countably infinite set V, whose element we call *variables*.

Definition 1.1. A *simple type* is a propositional formula in the language $\{\to\}$.

Definition 1.2. A *λ-term* is a string defined by the grammar $M \coloneqq x \mid M\,M \mid (\lambda x\, M)$. We denote by Λ the set of λ-terms. The set of *free variables* of a λ-term M is defined inductively by

$$FV(x) \coloneqq \{x\}\,, \quad FV(\lambda x\, M) \coloneqq FV(M) \setminus \{x\}\,, \quad FV(MN) \coloneqq FV(M) \cup FV(N).$$

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 τ_i is a simple type. If $(x : \tau) \in \Gamma$, we write $\Gamma(x) = \tau$, and we let

$$\operatorname{dom}\Gamma \coloneqq \{x \in V : (x : \tau) \text{ for some type } \tau\} \quad \text{and} \quad \operatorname{im}\Gamma \coloneqq \{\tau \text{ 'type'} : (x : \tau) \in \Gamma \text{ for some } x \in V\}\,.$$

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.

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

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*[2] $\Gamma \vdash M : \tau$.

(1) If $M = x$, then $\Gamma(x) = \tau$.
(2) If $M = PQ$, then $\Gamma \vdash P : \sigma \to \tau$ and $\Gamma \vdash Q : \sigma$ for some type σ.

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

(3) If $M = \lambda x\, N$ and $x \notin \operatorname{dom}\Gamma$, then $\tau = \tau_1 \to \tau_2$ and $\Gamma, x : \tau_1 \vdash N : \tau_2$ for some types τ_1, τ_2.

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 $\Gamma, x' : \tau_1 \vdash N' : \tau_2$, where $\lambda x\, N = \lambda x'\, N'$. Clearly $\tau = \tau_1 \to \tau_2$. Moreover, note that $N' = N[x'/x]$, so $\Gamma, x' : \tau_1 \vdash N[x'/x] : \tau_2$, and finally substituting x for x' back gives $\Gamma, x : \tau_1 \vdash N : \tau_2$, 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 $M = PQ$, then by Lemma 1.6.2, we have $\Gamma \vdash P : \sigma \to \tau$ and $\Gamma \vdash Q : \sigma$ for some type σ. 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 $M = \lambda x\, N$, we can choose $x \notin \operatorname{dom}\Gamma \cup \operatorname{dom}\Gamma'$, so that $\tau = \tau_1 \to \tau_2$ and $\Gamma, x : \tau_1 \vdash N : \tau_2$ by Lemma 1.6.3. By induction, we see that $\Gamma', x : \tau_1 \vdash N : \tau_2$, 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 $M = PQ$, 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 type ρ. By induction, we see that $\Gamma \vdash P[N/x] : \rho \to \tau$ and $\Gamma \vdash Q[N/x] : \rho$, on which App gives $\Gamma \vdash M[N/x] : \tau$.

If $M = \lambda y\, M'$ where $y \notin \operatorname{dom}\Gamma \cup \{x\} \cup FV(N)$, then by Lemma 1.6.3, there are types τ_1, τ_2 such that $\tau = \tau_1 \to \tau_2$ and $\Gamma, x : \sigma, y : \tau_1 \vdash M' : \tau_2$. By Lemma 1.7, we can weaken $\Gamma \vdash N : \sigma$ to $\Gamma, y : \tau_1 \vdash N : \sigma$, so by induction[3] we have $\Gamma, y : \tau_1 \vdash M'[N/x] : \tau_2$, 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 $\lambda x\, M \succ \lambda x\, N$ for each $x \in V$.

Definition 1.10. The least compatible relation $\to_\beta$ on Λ such that $(\lambda x\, M)N \to_\beta M[N/x]$ for all $M, N \in \Lambda$ is called *β-reduction*. We say that $(\lambda x\, M)N$ 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. In the case that $M = (\lambda x\, P)Q$ and $N = P[Q/x]$ for $x \notin \operatorname{dom}\Gamma$, there exist by Lemma 1.6.2 and 1.6.3 a term σ such that $\Gamma, x : \sigma \vdash P : \tau$ and $\Gamma \vdash Q : \sigma$, so $\Gamma \vdash N : \tau$ by Lemma 1.8. The general case follows by induction on $\twoheadrightarrow_\beta$, since the above describes a generic one-step β-reduction. ∎

2. The polymorphic λ-calculus: System F

Definition 2.1.

Lemma 2.2.

Theorem 2.3 (Subject Reduction for **F**).

3. The λ-calculus with Dependent Types: λP

Definition 3.1.

Lemma 3.2.

Theorem 3.3 (Subject Reduction for λ**P**).

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

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

Definition 4.1.

Lemma 4.2.

Theorem 4.3 (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