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 countably infinite sets V_p and V_t, whose elements we call *propositional variables* and *type variables*, respectively. Since this is the most basic (typed) λ-calculus, we will be very formal here.

Definition 1.1. A *simple type* is a propositional formula in the language $\{\to\}$; i.e., defined by the grammar $T \coloneqq V_t \mid T \to T$. We generally use the letters $\tau, \sigma, \ldots \in T$ for types.

Definition 1.2. A *λ-term* is defined by the grammar $\Phi \coloneqq V_p \mid \Phi\,\Phi \mid \lambda V_p : T.\Phi$. We generally use the letters $M, N, \ldots \in \Phi$ to denote λ-terms.

Definition 1.3. A *judgement* is a pair $(M, \tau) \in \Phi \times T$, denoted $M : \tau$.

Remark 1.4. 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 1.5. A *context* is a finite set of pairs $\Gamma \coloneqq \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$, where $x_i \in V_p$ are pairwise-distinct and each $\tau_i \in T$; that is, $\Gamma : V_p \rightharpoonup T$ is a partial function, so we write $\Gamma(x) = \tau$ for $(x : \tau) \in \Gamma$ and we let

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

Definition 1.6. Fix a context Γ. We say that a judgement $M : \tau$ is *$\lambda_\to$-derivable from* Γ, and write $\Gamma \vdash M : \tau$, if there is a finite tree of judgements rooted at $M : \tau$, whose leaves are instances of Init, and such that the children of each internal node is obtained from the rules Abs or App read bottom-up.

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

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

Example 1.7. We have $\vdash (\lambda x : \tau.x) : (\tau \to \tau)$; $\vdash (\lambda x : \tau.(\lambda y : \sigma.x)) : (\tau \to \sigma \to \tau)$; and $x : \tau \vdash (\lambda y : \sigma.x) : (\sigma \to \tau)$.

Lemma 1.8 (Generation Lemma)**.** *Suppose that* $\Gamma \vdash M : \tau$.

Date: April 10, 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'.

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

Proof. Since the root of the derivation tree for $\Gamma \vdash M : \tau$ determines the shape of M, we see that (1) follows from Init and (2) follows from App. We defer the proof of (3) until after Lemma 1.11. □

Definition 1.9. The set of *free variables* of a λ-term $M \in \Phi$ is defined inductively by

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

Definition 1.10. Let $M, N \in \Phi$ and fix $x \in FV(M)$. The *substitution* of N for x in M, written $M[N/x]$, is the λ-term defined by induction on M; below, $y \neq x$ and $y \notin FV(N)$.

$$x[N/x] \coloneqq N, \quad y[N/x] \coloneqq y, \quad (P\,Q)[N/x] \coloneqq P[N/x]\,Q[N/x], \quad (\lambda y : \tau . P)[N/x] \coloneqq \lambda y : \tau . P[N/x].$$

Lemma 1.11 (Variable Substitution)**.** *If $\Gamma, x : \tau \vdash M : \sigma$ and $y \notin \operatorname{dom}\Gamma \cup \{x\}$, then $\Gamma, y : \tau \vdash M[y/x] : \sigma$.*

Proof. By induction on the length of M. If $M = x$, then $\tau = \sigma$ and $M[y/x] = y$, and $\Gamma, y : \tau \vdash y : \sigma$ by Init. If $M = z$ and $z \neq x$, then $M[y/x] = z$. Note that $\Gamma(z) = \sigma$ by Lemma 1.8.1, so $\Gamma, y : \tau \vdash z : \sigma$ by Init.

If $M = P\,Q$, then by Lemma 1.8.2, we have $\Gamma, x : \tau \vdash P : \rho \to \sigma$ and $\Gamma, x : \tau \vdash Q : \rho$ for some $\rho \in T$. By induction, we have $\Gamma, y : \tau \vdash P[y/x] : \rho \to \sigma$ and $\Gamma, y : \tau \vdash Q[y/x] : \rho$, on which App gives the desired.

If $M = \lambda z : \rho . N$, then $\Gamma, x : \tau \vdash \lambda z : \rho . N : \sigma$ is obtained by Abs, so it is of the form $\Gamma, x : \tau, w : \rho \vdash P : \eta$ for some $\eta \in T$ and $w, P \in \Phi$ such that $\lambda z : \rho . N = \lambda w : \rho . P$ and $\sigma = \rho \to \eta$. Up to α-conversion, we can choose w so that $w \neq y$. Then, since $|P| = |N| < |M|$, we have by induction that $\Gamma, y : \tau, w : \rho \vdash P[y/x] : \eta$, on which Abs gives $\Gamma, y : \tau \vdash (\lambda w : \rho . P[y/x]) : \sigma$. Now, $\lambda w : \rho . P[y/x] = (\lambda w : \rho . P)[y/x] = (\lambda z : \rho . N)[y/x] = M[y/x]$. ■

Proof of Lemma 1.8.3. The proof $\Gamma \vdash (\lambda x : \sigma . N) : \tau$ must be obtained from Abs, so there is some $\rho \in T$ such that $\tau = \sigma \to \rho$ and some $y, P \in \Phi$ such that $\Gamma, y : \sigma \vdash P : \rho$ and $\lambda x : \sigma . N = \lambda y : \sigma . P$. If $x = y$, then $N = P$ and we are done. Otherwise, we have $P = N[x/y]$, so $\Gamma, y : \sigma \vdash N[x/y] : \rho$, so substituting y for x and using Lemma 1.11 gives $\Gamma, x : \sigma \vdash N : \rho$, as desired. ■

Lemma 1.12 (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 1.8.1, and hence $\Gamma' \vdash x : \tau$ by Init. If $M = P\,Q$, then by Lemma 1.8.2, we have $\Gamma \vdash P : \sigma \to \tau$ and $\Gamma \vdash Q : \sigma$ for some $\sigma \in T$. 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 1.8.3. 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 1.13 (Substitution Lemma)**.** *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 Init. 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 1.8.2, we have $\Gamma, x : \sigma \vdash P : \rho \to \tau$ and $\Gamma, x : \sigma \vdash Q : \rho$ for some $\rho \in T$. 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 1.8.3, there is some $\rho \in T$ such that $\tau = \eta \to \rho$ and $\Gamma, x : \sigma, y : \eta \vdash M' : \rho$. By Lemma 1.12, we can weaken $\Gamma \vdash N : \sigma$ to $\Gamma, y : \eta \vdash N : \sigma$, so by induction[2] we have $\Gamma, y : \eta \vdash M'[N/x] : \rho$, and we can apply Abs to get $\Gamma \vdash M[N/x] : \tau$. ■

Definition 1.14. A relation $\succ$ on Φ is *compatible* if for any $M, N \in \Phi$ with $M \succ N$, we have $MP \succ NP$ and $PM \succ PN$ for each $P \in \Phi$, and $\lambda x : \tau . M \succ \lambda x : \tau . N$ for each $x \in V$ and $\tau \in T$.

Definition 1.15. The least compatible relation $\to_\beta$ on Φ satisfying $(\lambda x : \tau . M)N \to_\beta M[N/x]$ for all $M, N \in \Phi$ is called *β-reduction*. The term $(\lambda x : \tau . M)N$ is a *β-redex*, and $M[N/x]$ arises by *contracting* the redex.

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

Notation 1.16. 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 1.17 (Subject Reduction)**.** *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 1.8.2 and 1.8.3, so $\Gamma \vdash N : \tau$ by Lemma 1.13. The general case follows by induction on $\twoheadrightarrow_\beta$. ∎

In summary, the *simply-typed λ-calculus*, denoted $\lambda_\to$, consists of the following data:

(1) A set $T ::= V_t \mid T \to T$ of *λ-types* and a set $\Phi ::= V_p \mid \Phi\,\Phi \mid \lambda V_p : T . \Phi$ of *λ-terms*.
(2) A *β-reduction rule* $\to_\beta$ on Φ compatibly extending $(\lambda x : \tau . M)N \to_\beta M[N/x]$ for all $M, N \in \Phi$.
(3) A *type assignment* relation $\Gamma \vdash M : \tau$ defined by Init, App, and Abs.

All other λ-calculi considered here will follow a similar specification, so we will be briefer in what follows.

2. The polymorphic λ-calculus: λ2

Definition 2.1. The *polymorphic λ-calculus*, denoted λ2, consists of the following data.

(1) A set $T ::= V_t \mid T \to T \mid \forall V_t . T$ of *polymorphic types*.
(2) A set $\Phi ::= V_p \mid \Phi\,\Phi \mid \lambda V_p : T . \Phi \mid \Lambda V_t . \Phi \mid \Phi\,T$ of *polymorphic terms*.
(3) A *β-reduction* rule on Φ compatibly extending $(\lambda x : \tau . M)N \to_\beta M[N/x]$ and $(\Lambda p . M)\tau \to_\beta M[\tau/p]$ for all $M, N \in \Phi$ and $\tau \in T$.
(4) A *type assignment* relation $\Gamma \vdash M : \tau$ defined by Init, App, Abs, and

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash \Lambda p . M : \forall p . \tau}\ \textsc{Gen}\ (\text{if } p \notin FV(\Gamma)) \qquad \frac{\Gamma \vdash M : \forall p . \sigma}{\Gamma \vdash M\,\tau : \sigma[\tau/p]}\ \textsc{Inst}$$

A moments thought reveals that all results in the previous section apply here. Indeed, the quantifier ∀ is introduced and eliminated independently from that of →, and all notions regarding → is left unchanged.

Notation 2.2. We generally let $x, y, z, \ldots \in V_p$, $p, q, r, \ldots \in V_t$, $\tau, \sigma, \rho, \ldots \in T$, and $M, N, P, \ldots \in \Phi$. Also, note that α-conversion applies to the Λ-binding too, so we can freely change p in $\Lambda p . M$.

Lemma 2.3 (Variable Substitution)**.** *If $\Gamma \vdash M : \tau$ and $\sigma \in T$, then $\Gamma[\sigma/p] \vdash M[\sigma/p] : \tau[\sigma/p]$.*

Lemma 2.4 (Generation Lemma)**.** *Suppose that $\Gamma \vdash M : \tau$.*

(1) If $M = x$, $M = P\,Q$, or $M = \lambda x : \sigma . N$, then Lemma 1.8 applies.
(2) If $M = N\,\sigma$, then $\tau = \rho[\sigma/p]$ and $\Gamma \vdash N : \forall p . \rho$ for some $\rho \in T$.
(3) If $M = \Lambda p . N$ and $p \notin FV(\operatorname{ran}\Gamma)$, then $\tau = \forall p . \sigma$ and $\Gamma \vdash N : \sigma$ for some $\sigma \in T$.

Lemma 2.5 (Change of Context)**.** *If $\Gamma \vdash M : \tau$ and $\Gamma(x) = \Gamma'(x)$ for all $x \in FV(M)$, then $\Gamma' \vdash M : \tau$.*

Lemma 2.6 (Substitution Lemma)**.** *If $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma$, then $\Gamma \vdash M[N/x] : \tau$.*

Theorem 2.7 (Subject Reduction)**.** *If $\Gamma \vdash M : \tau$ and $M \twoheadrightarrow_\beta N$, then $\Gamma \vdash N : \tau$.*

Proof. By Theorem 1.17 (and Lemma 2.6), it suffices to prove it for when $M = (\Lambda p . P)\sigma$ and $N = P[\sigma/p]$ for some $P \in \Phi$ and $\sigma \in T$. After an α-conversion, we can assume without loss of generality that $p \notin FV(\operatorname{ran}\Gamma)$, so by Lemma 2.4, we have $\tau = \rho[\sigma/p]$ and $\Gamma \vdash P : \rho$ for some $\rho \in T$, and hence $\Gamma[\sigma/p] \vdash N : \tau$ by Lemma 2.3. But $(\Gamma[\sigma/p])(x) = \Gamma(x)[\sigma/p] = \Gamma(x)$ since $p \notin FV(\operatorname{ran}\Gamma)$, so the result follows from Lemma 2.5. ∎

TODO: someone check this please

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

Definition 3.1.

Lemma 3.2.

Theorem 3.3 (Subject Reduction)**.**

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

Definition 4.1.

Lemma 4.2.

Theorem 4.3 (Subject Reduction)**.**

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

Definition 5.1.

Lemma 5.2.

Theorem 5.3 (Subject Reduction)**.**

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