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 element 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 $\tau, \sigma, \rho, \ldots \in T$ for types.

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

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 of pairs $\Gamma \coloneqq \{x_1\!:\!\tau_1, \ldots, x_n\!:\!\tau_n\}$, where $x_i \in V$ are pairwise-distinct and each $\tau_i \in T$; that is, $\Gamma : V \rightharpoonup T$ is a partial function, so we write $\Gamma(x) = \tau$ for $(x\!:\!\tau) \in \Gamma$ and we let

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

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 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 $x \notin \operatorname{dom}\Gamma$. We assert $\Gamma \vdash M\!:\!\tau$ if it is derivable.

Definition 1.6. The set of *free variables* of a λ-term $M \in \Lambda$ is defined inductively by

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

Date: April 9, 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'.

Definition 1.7. Let $M, N \in \Lambda$ and fix $x \in FV(M)$. The *substitution* of N for x in M, written $M[N/x]$, is 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 1.8 (Generation Lemma 1)**.** *Suppose that $\Gamma \vdash M{:}\tau$.*

(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 $\sigma \in T$.

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. ■

Lemma 1.9 (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 $w, P \in \Lambda$ 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, $\lambda w{:}\rho.P[y/x] = (\lambda w{:}\rho.P)[y/x] = (\lambda z{:}\rho.N)[y/x] = M[y/x]$. ■

Lemma 1.10 (Generation Lemma 2)**.** *If $\Gamma \vdash \lambda x{:}\sigma.M{:}\tau$ and $x \notin \operatorname{dom}\Gamma$, then $\tau = \sigma \to \rho$ and $\Gamma, x{:}\sigma \vdash N{:}\rho$ for some $\rho \in T$.*

Proof. As in the above proof, there exist $\rho \in T$ and $y, N \in \Lambda$ such that $\Gamma, y{:}\sigma \vdash N{:}\rho$, $\lambda x{:}\sigma.M = \lambda y{:}\sigma.N$, and $\tau = \sigma \to \rho$. If $x = y$, we are done; otherwise, we have $N = M[y/x]$, so $\Gamma, y{:}\sigma \vdash M[y/x]{:}\rho$, and finally substituting x for y gives $\Gamma, x{:}\sigma \vdash M{:}\rho$ by Lemma 1.9, as desired. ■

Lemma 1.11 (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 1.10. 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 1.12 (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 1.10, there is some $\rho \in T$ such that $\tau = \eta \to \rho$ and $\Gamma, x{:}\sigma, y{:}\eta \vdash M'{:}\rho$. By Lemma 1.11, 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 1.13. 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 $\tau \in T$.

Definition 1.14. 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.

Definition 1.15. The *simply-typed λ-calculus* consists of the following data:

(1) A set T of *λ-types*, defined by $T \coloneqq V_t \,|\, T \to T$. We generally use the letters $\tau, \sigma, \rho, \ldots \in T$.

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

(2) A set Λ of *λ-terms*, defined by $\Lambda \coloneqq V_p \mid \Lambda\,\Lambda \mid \lambda V_p\,{:}\,T.\Lambda$. We generally use the letters $M, N, P, \ldots \in \Lambda$.
(3) A *β-reduction rule*, defined as the least compatible relation $\to_\beta$ on Λ such that $(\lambda x : \tau.M)N \to_\beta M[N/x]$ for all $M, N \in \Lambda$.
(4) A *type assignment* relation $\vdash$ on triples (Γ, M, τ) where $\Gamma : V_p \rightharpoonup \Lambda$ is a partial function (called a *context*), $M \in \Lambda$, and $\tau \in T$, defined axiomatically by the following rules

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 1.10, so $\Gamma \vdash N\,{:}\,\tau$ by Lemma 1.12. The general case follows by induction on $\twoheadrightarrow_\beta$. ■

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

Throughout, fix two disjoint countable sets V and V_t of variables and *type-variables*.

Definition 2.1. A *polymorphic type* is a propositional formula in the language $\{\to, \forall\}$ over V_τ, i.e., defined by the grammar $\tau \coloneqq p \mid \tau \to \tau \mid \forall p\, \tau$ where $p \in V_t$. Let Φ_2 denote the set of polymorphic types.

Definition 2.2. A *polymorphic term* is either a λ-term, a *polymorphic abstraction* $\Lambda p\, M$ for $p \in V_t$, or a *type application* $M\,\tau$ for $\tau \in \Phi_2$. That is, $M \coloneqq x \mid M\,M \mid \lambda x : \tau.M \mid \Lambda p\, M \mid M\,\tau$ where $\tau \in \Phi_2$ and $p \in V_t$. We denote by Λ_2 the set of all polymorphic terms. The set of *free variables* of a polymorphic term M and a polymorphic type τ is defined inductively by extending FV from before, and letting

$$FV(\Lambda p\, M) \coloneqq FV(M) \setminus \{p\}\,, \qquad FV(M\,\tau) \coloneqq FV(M) \cup FV(\tau)$$
$$FV(p) \coloneqq \{p\}\,, \qquad FV(\tau \to \sigma) \coloneqq FV(\tau) \cup FV(\sigma), \qquad FV(\forall p\, \tau) \coloneqq FV(\tau) - \{p\}\,.$$

Definition 2.3. Let $\tau, \sigma \in \Phi_2$ and fix $p \in FV(\tau)$. The *substitution* of σ for p in τ, written $\tau[\sigma/p]$, is the polymorphic type defined by induction on τ; below, $q \neq p$ and $q \notin FV(\sigma)$.

$$p[\sigma/p] \coloneqq \sigma, \qquad q[\sigma/p] \coloneqq q, \qquad (\tau_1 \to \tau_2)[\sigma/p] \coloneqq \tau_1[\sigma/p] \to \tau_2[\sigma/p], \qquad (\forall q\, \tau)[\sigma/p] \coloneqq \forall q\, \tau[\sigma/p].$$

If $\Gamma : V \rightharpoonup \Phi_2$ is a context, we let $(\Gamma[\sigma/p])(x) \coloneqq \Gamma(x)[\sigma/p]$.

Definition 2.4. For $M, N \in \Lambda_2$ and $x \in FV(M)$, we extend the substitution $M[N/x]$ by letting $(\Lambda p\, M)[N/x] \coloneqq \Lambda p\, M[N/x]$ for $p \notin FV(N)$ and $(M\,\tau)[N/x] \coloneqq M[N/x]\,\tau$. For $p \in FV(M)$ and $\sigma \in \Phi_2$, we define the *substitution* of σ for p in M, written $M[\sigma/p]$, as the polymorphic term below, where $q \notin FV(\sigma) \cup \{p\}$.

$$x[\sigma/p] \coloneqq x, \qquad (P\,Q)[\sigma/p] \coloneqq P[\sigma/p]\,Q[\sigma/p], \qquad (\lambda x\,{:}\,\tau.M)[\sigma/p] \coloneqq \lambda x\,{:}\,\tau[\sigma/p].M[\sigma/p],$$
$$(M\,\tau)[\sigma/p] \coloneqq M[\sigma/p]\,\tau[\sigma/p], \qquad (\Lambda q\, M)[\sigma/p] \coloneqq \Lambda q\, M[\sigma/p].$$

Definition 2.5. A judgement $\Gamma \vdash M\,{:}\,\tau$ is *derivable in* $\lambda 2$ if it is derivable in $\lambda_\to$ with the additional rules:

$$\frac{\Gamma \vdash M\,{:}\,\tau}{\Gamma \vdash \Lambda p\, M\,{:}\,\forall p\, \tau}\ \textsc{Gen} \qquad \frac{\Gamma \vdash M\,{:}\,\forall p\, \sigma}{\Gamma \vdash M\,\tau\,{:}\,\sigma[\tau/p]}\ \textsc{Inst}$$

The Gen rule can only be applied if $p \notin FV(\Gamma)$.

A moments thought reveals that all results in the previous section apply here, with the only change that:

Notation 2.6. Throughout, we let $x, y, \ldots \in V$, $p, q, \ldots \in V_t$, $\tau, \sigma, \ldots \in \Phi_2$, and $M, N, \ldots \in \Lambda_2$. Also, note that α-conversion applies to the Λ-binding too, so we can freely change p in $\Lambda p\, M$.

Lemma 2.7 (Variable Substitution)**.** *If $\Gamma \vdash M\,{:}\,\tau$, then $\Gamma[\sigma/p] \vdash M[\sigma/p]\,{:}\,\tau[\sigma/p]$.*

Proof. **TODO** ■

Lemma 2.8 (Generation Lemma 1)**.** *If $\Gamma \vdash M\,\tau\,{:}\,\sigma$, then $\sigma = \rho[\tau/p]$ and $\Gamma \vdash M\,{:}\,\forall p\, \rho$.*

Proof. **TODO** ■

Lemma 2.9 (Generation Lemma 2)**.** *If $\Gamma \vdash \Lambda p\, M\,{:}\,\tau$ and $p \notin FV(\operatorname{ran}\Gamma)$, then $\tau = \forall p\, \sigma$ and $\Gamma \vdash M\,{:}\,\sigma$.*

Proof. **TODO** ■

Lemma 2.10 (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. **TODO** ■

Lemma 2.11 (Substitution Lemma)**.** *If $\Gamma, x : \sigma \vdash M : \tau$ and $\Gamma \vdash N : \sigma$, then $\Gamma \vdash M[N/x] : \tau$.*

Proof. **TODO** ■

Definition 2.12. The least compatible relation $\to_\beta$ on Λ_2 such that $(\lambda x : \tau . M)N \to_\beta M[N/x]$ and $(\Lambda p\, M)\tau \to_\beta M[\tau/p]$ is called *β-reduction*. The same terminology applies.

Theorem 2.13 (Subject Reduction)**.** *If $\Gamma \vdash M : \tau$ and $M \twoheadrightarrow_\beta N$, then $\Gamma \vdash N : \tau$.*

Proof. By Theorem 1.17, it suffices to prove it for when $M = (\Lambda p\, P)\sigma$ and $N = P[\sigma/p]$. Assuming without loss of generality that $p \notin FV(\operatorname{ran}\Gamma)$, we have by Lemmas 2.8 and 2.9 that $\tau = \rho[\sigma/p]$ and $\Gamma \vdash P : \rho$ for some ρ, whence $\Gamma[\sigma/p] \vdash N : \tau$ by Lemma 2.7. But note that $(\Gamma[\sigma/p])(x) = \Gamma(x)[\sigma/p] = \Gamma(x)$ since $p \notin FV(\operatorname{ran}\Gamma)$, so the result follows from Lemma 2.10. **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