FUNCTIONAL PEARLS
α-conversion is easy

THORSTEN ALTENKIRCH
School of Computer Science & Information Technology, University of Nottingham, Nottingham, NG8 1BB, UK
(*e-mail:* txa@cs.nott.ac.uk)

Abstract

We present a new and simple account of α-conversion suitable for formal reasoning. Our main tool is to define α-conversion as a a structural congruence parametrized by a partial bijection on free variables. We show a number of basic properties of substitution. e.g. that substitution is monadic which entails all the usual substitution laws. Finally, we relate α-equivalence classes to de Bruijn terms.

1 Introduction

When reasoning about λ-terms we usually want to identify terms which only differ in the names of bound variables, such as $\lambda x.zx$ and $\lambda y.zy$. We say that these terms are α-convertible and write $\lambda x.zx \equiv_\alpha \lambda y.zy$. We want that all operations on λ-terms preserve $\equiv_\alpha$. The first potential problem is substitution, if we naively substitute z by x in the terms above we obtain $(\lambda x.zx)[z \leftarrow x] = \lambda x.xx$ and $(\lambda y.zy)[z \leftarrow x] = \lambda y.xy$ but $\lambda x.xx \not\equiv_\alpha \lambda y.xy$. To avoid this behaviour we introduce *capture avoiding substitution*:

$$(\lambda x.t)[y \leftarrow u] = \lambda x'.t[x \leftarrow x'][y \leftarrow u] \qquad \text{where } x' \text{ does not occur free in } u \text{ or } t.$$

in the case $x \neq y$.

This solves the problem discussed before: $(\lambda x.zx)[z \leftarrow x] = \lambda v.xv$ and $(\lambda y.zy)[z \leftarrow x] = \lambda v'.xv'$ and indeed $\lambda v.xv \equiv_\alpha \lambda v'.xv'$. Here v, v' are arbitrary choices of variable names — we can make sure that such variables exist by using a countably infinite set of names.

How do we establish formally that the improved definition of substitution preserves α-conversion and has some other desirable properties? The standard account (Curry & Feys, 1958; Hindley & Seldin, 1986) defines α-conversion as structural, transitive closure of

$$\frac{y \notin \mathrm{FV}(t)}{\lambda x.t \equiv_\alpha \lambda y.t[x \leftarrow y]]}$$

and establish basic properties[1] such as symmetry and that substitution preserves α equivalence. (Hindley & Seldin, 1986), p. 10, remark:

The above technicalities are rather dull. But their very dullness has made them a trap for even the most careful authors.

One possible reaction to the unexpected complexity of something as apparently trivial as α-conversion is to ignore it. Indeed, this was the approach initially taken in (Barendregt, 1984), only the 2nd edition contains a discussion of the topic in the appendix (pp. 577).

Another reaction is to look for an alternative presentation of λ-terms. The classical approach are de Bruijn's *nameless dummies* (de Bruijn, 1972). This approach is suitable for formalisation, e.g. see (Altenkirch, 1993), but can get quite unwieldy in some cases. Also it leaves open the question how de Bruijn's terms are related to named terms. Alternatively, Coquand suggested to introduce two different classes of variables (bound variables and parameters), this theory is formally developed in (McKinna & Pollack, 1993).

Yet another possibility is to move to a world where the problem disappears, this is basically the idea behind recent work on *higher order abstract syntax* and related approaches (Hofmann, 1999; Gabbay & Pitts, 1999; Fiore *et al.*, 1999).

Somewhere orthogonal to this, Gordon and Melham suggested and investigated an axiomatic theory to reason about α-conversion (Gordon & Melham, 1996) — this has been formalized in HOL.

In the present note I attempt something maybe less exciting but hopeful as least as useful: I suggest a better (I hope) way to present and reason about the classical theory of α-conversion.

The main tool is to define a generalized version of α-conversion which also deals with free variables and to use techniques from typed λ-calculus to reason about it. This definition of substitution fits into the structure of a monad and by verifying that it satisfies the equational properties of a monad (or a Kleisli triple) we are able to show that the usual substitution laws hold. As a corollary we obtain that finite sets of variables with substitutions as morphisms form a category. Finally, we will discuss the relation to de Bruijn terms.

2 Terms and α-conversion

We assume as given an infinite set of variable names $\mathcal{V}$ with a decidable equality. Infinity guarantees that for any finite set of names $X \subset_{\text{fin}} \mathcal{V}$ we can effectively obtain $\text{Fresh}(X) \in \mathcal{V} \setminus X$.

Instead of defining a set of terms we define a family of sets $\text{Tm}(X)$ of terms with free variable in $X \subseteq_{\text{fin}} \mathcal{V}$ inductively:

$$\frac{x \in X}{x \in \text{Tm}(X)} \qquad \frac{t, u \in \text{Tm}(X)}{t\,u \in \text{Tm}(X)} \qquad \frac{t \in \text{Tm}(X \cup \{x\})}{\lambda x.t \in \text{Tm}(X)}$$

[1] As far as I am aware it is never shown that transitivity can be eliminated from the definition of α-conversion, which is needed to show that the relation is non-trivial.

This definition is motivated by the view that the untyped λ-calculus is a special case of the typed one, where one only has got one type. Hence the judgment $\Gamma \vdash t : \sigma$ turns into $t \in \mathrm{Tm}(X)$ since σ is unique and all what remains from the context Γ is the set of names occurring in it.

The basic problem with the classical definition of α-conversion is that it doesn't take free variables into account — it only talks about bound variables. However, when moving under a binder previously bound variables become free. Our solution is to parametrize α-conversion by a partial bijection — a partial bijection $R \subseteq X \times Y$ on finite sets of variables $X, Y \subset_{\mathrm{fin}} \mathcal{V}$ is a relation which is a partial function and its converse is a partial function. Or with other words it is a bijection on a subset of X and Y.

Given R a partial bijection as above and $x, y \in \mathcal{V}$ we define the symmetric update of R as

$$R(x,y) = (R \setminus \{(x,v),(y,w) \mid v,w \in \mathcal{V}\}) \cup \{(x,y)\} \quad \in (X \cup \{x\}) \times (Y \cup \{y\})$$

It is easy to see that $R(x,y)$ is a partial bijection.

We now define $\equiv_\alpha^R \subseteq \mathrm{Tm}(X) \times \mathrm{Tm}(Y)$ parametrized by a partial bijection $R \subseteq X \times Y$, inductively:

$$\frac{xRy}{x \equiv_\alpha^R y} \qquad \frac{t \equiv_\alpha^R t' \quad u \equiv_\alpha^R u'}{t\,u \equiv_\alpha^R t'\,u'} \qquad \frac{t \equiv_\alpha^{R(x,y)} u}{\lambda x.t \equiv_\alpha^R \lambda y.u}$$

Note that we cannot replace partial bijections by bijections because of shadowing, e.g. $(\lambda x.x)x \equiv_\alpha (\lambda y.y)x$ cannot be established using only bijections.

Given $X, Y, Z \subset_{\mathrm{fin}} \mathcal{V}$ we write $1_X = \{(x,x) \mid x \in X\} \subseteq X \times X$ for the identity relation on X, which obviously is a partial bijection. Given $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ we write $R^\circ = \{(y,x) \mid (x,y) \in R\} \subseteq Y \times X$ for R's converse and $R;S = \{(x,z) \mid \exists y \in Y.(x,y) \in R \wedge (y,z) \in S\} \subseteq X \times Z$ for the relational composition of S and Z. Both operations are closed under partial bijections. We can now establish basic properties of update:

Lemma 1
Let $X, Y, Z \subset_{\mathrm{fin}} \mathcal{V}$ and $R \subseteq X \times Y, S \subseteq Y \times Z$ be partial bijections:

1. $1_X(x,x) = 1_{X \cup \{x\}}$
2. $(R(x,y))^\circ = R^\circ(y,x)$
3. $R(x,y);S(y,z) = (R;S)(x,z)$

Proof
I show only 3. the other cases are even easier:

$$\begin{array}{ll} & aR(x,y);S(y,z)b \\ \iff & \quad \{ \text{ definition of update and } ; \ \} \\ & \exists c.(a = x \wedge y = c \wedge z = b) \vee (x \neq a \wedge y \neq c \wedge z \neq b \wedge aRc \wedge cSb \\ \iff & \quad \{ \text{ properties of } \exists, \text{ defn of } ; \ \} \\ & (a = x \wedge z = b) \vee (x \neq a \wedge z \neq b \wedge aR;Sb) \\ \iff & \quad \{ \text{ definition of update } \} \\ & a(R;S)(x,z)b \end{array}$$

□

Proposition 2

1. $\forall t \in \mathrm{Tm}(X).t \equiv_\alpha^{1_X} t$
2. $t \equiv_\alpha^R u \implies u \equiv_\alpha^{R^\circ} t$
3. $t \equiv_\alpha^R u \wedge u \equiv_\alpha^S v \implies t \equiv_\alpha^{R;S} u$

Proof
By induction over the structure of $t \in \mathrm{Tm}(X)$ using the fact that the rules defining $\equiv_\alpha^R$ are structurally deterministic. The only difficult case is λ where we have to use the corresponding parts of lemma 1. □

We now define $\equiv_\alpha = \equiv_\alpha^{1_X}$ and obtain

Corollary 3
$\equiv_\alpha$ is an equivalence relation, i.e. it is reflexive, symmetric, transitive.

Proof
Directly from the previous proposition. □

Since we are interested in terms up to α-equivalence, we define the family of sets of α-equivalence classes $\mathrm{Tm}^\alpha(X) = \mathrm{Tm}(X)/\equiv_\alpha$.

3 Substitution

A substitution is given by a function $f \in X \to \mathrm{Tm}(Y)$ where $X, Y \subset_{\mathrm{fin}} \mathcal{V}$. Given a substitution f and $x \in \mathcal{V}, t \in \mathrm{Tm}(Y)$ we define the update

$$\begin{aligned} f[x,t] &\in X \cup \{x\} \to \mathrm{Tm}(Y) \\ &= f \setminus \{(x,u) \mid u \in \mathrm{Tm}(Y)\} \cup \{(x,t)\} \end{aligned}$$

A substitution can be extended to a function on terms $(\!|f|\!) \in \mathrm{Tm}(X) \to \mathrm{Tm}(Y)$, this extension proceeds by structural recursion over $t \in \mathrm{Tm}(X)$:

$$\begin{aligned} (\!|f|\!)(x) &= f(x) \\ (\!|f|\!)(t\ u) &= (\!|f|\!)(t)\ (\!|f|\!)(u) \\ (\!|f|\!)(\lambda x.t) &= \lambda z.(\!|f[x,z]|\!)(t) \qquad \text{where } z = \mathrm{Fresh}(Y) \end{aligned}$$

The common case of substituting a single variable arises as a a special case, i.e. given $x \in \mathcal{V}$ and $u \in \mathrm{Tm}(X)$ we have $1_X[x,u] \in X \cup \{x\} \to X$. Hence $(\!|1_X[x,u]|\!) \in \mathrm{Tm}(X \cup \{x\}) \to \mathrm{Tm}(X)$ is the operation which replaces x by u. Given $t \in \mathrm{Tm}(X \cup \{x\})$ we write $t[x=u]$ for $(\!|1_X[x,u]|\!)(t)$.

We are now going to verify that substitution preserves α-congruence, i.e. if we have two substitutions $f, g \in X \to \mathrm{Tm}(Y)$ s.t. $f(x) \equiv_\alpha g(x)$ then for any $t \equiv_\alpha u$ we have $(\!|f|\!)(t) \equiv_\alpha (\!|g|\!)(u)$.

We use the usual notation to extend relations to function spaces, i.e. $fR \to Sg \iff \forall x, x'.xRx' \implies f(x)Sg(x')$. Using this we can express the preservation theorem in a general form:

Proposition 4
Given $f \in X \to \mathrm{Tm}(Y), g \in X' \to \mathrm{Tm}(Y')$ and partial bijections $R \subseteq X \times X', S \subseteq Y \times Y'$ we have that

$$\frac{fR \to (\equiv_\alpha^S)g}{(\!|f|\!)(\equiv_\alpha^R) \to (\equiv_\alpha^S)(\!|g|\!)}$$

Clearly, the preservation property arises as a special case by setting $R = 1_X$ and $S = 1_Y$.

To prove this we first need a lemma on updates:

Lemma 5
Given the assumptions of prop. 4 and $z \notin Y, z' \notin Y'$ we have

$$\frac{fR \to Sg}{f[x,z]R(x,y) \to \equiv_\alpha^{S(z,z')} g[y,z']}$$

Proof
Simple case analysis on $vR(x,y)v'$. □

Proof of prop. 4
The proposition is equivalent to saying that if $t \equiv_\alpha^R u$ then for all f, g, S if $fR \to (\equiv_\alpha^S)g$ then $f(t) \equiv_\alpha^S g(u)$. We proceed by induction over the derivation of $t \equiv_\alpha^R u$.

The only difficult case (as usual) is λ. Assume we have derived $\lambda x.t \equiv_\alpha^R \lambda y.u$ from $t \equiv_\alpha^{R(x,y)} u$. Let $z = \mathrm{Fresh}(Y)$ and $z' = \mathrm{Fresh}(Y')$.

We assume $fR \to (\equiv_\alpha^S)g$, hence by lemma 5 we have

$$f[x,z]R(x,y) \to \equiv_\alpha^{S(z,z')} g[y,z']$$

Hence by ind.hyp. we know $(\!|f[x,z]|\!)(t) \equiv_\alpha^{S(z,z')} (\!|f[y,z']|\!)(u)$ and

$$\begin{array}{rcl} (\!|f|\!)(\lambda x.t) & = & \lambda z.(\!|f[x,z]|\!)(t) \\ & \equiv_\alpha^S & \lambda z'.(\!|f[y,z']|\!)(u) \\ & = & (\!|g|\!)(\lambda y.u) \end{array}$$

□

A consequence of proposition 4 is that substitution is an operation on α-equivalence classes, that is given $f \in X \to \mathrm{Tm}^\alpha(Y)$ then $(\!|f|\!) \in \mathrm{Tm}^\alpha(X) \to \mathrm{Tm}^\alpha(Y)$.

4 Substitution is monadic

To show that substitution is well behaved, i.e. laws such as

$$t[x \leftarrow u] = t \qquad \text{if } x \notin \mathrm{FV}(t) \tag{L1}$$

$$t[x \leftarrow u][y \leftarrow v] = t[y \leftarrow v][x \leftarrow u[y \leftarrow v]] \qquad \text{if } x \neq y \tag{L2}$$

hold, we establish that substitution is monadic:

Proposition 6
$(\mathrm{Tm}^\alpha, \eta, (\!|-|\!))$ is a monad, where the unit $\eta_X \in X \to \mathrm{Tm}^\alpha(X)$ is the embedding $\eta(x) = x$. That is, the following equations are satisfied:

1. $(\!|\eta_X|\!) = 1_{\mathrm{Tm}^\alpha(X)}$
2. $(\!|f|\!) \circ \eta = f$
3. $(\!|f|\!) \circ (\!|g|\!) = (\!|(\!|f|\!) \circ g|\!)$

where $g \in X \to \mathrm{Tm}^\alpha(Y), f \in Y \to \mathrm{Tm}^\alpha(Z)$.

Before proving prop. 6 let's see how it can be applied:

For L1 assume $t, u \in \mathrm{Tm}^\alpha(X)$ and $x \notin X$. Now $1_X[x, u] \in X \cup \{x\} \to \mathrm{Tm}^\alpha(X)$ and since $X \subseteq X \cup \{x\}$ also $1_X[x = u] \in X \to \mathrm{Tm}^\alpha(X)$. However, a simple case analysis shows that as function over X: $1_X[x = u] = \eta_X$ and using 1. we have $(\!|1_X[x, u]|\!)(t) = \eta_X(t) = t$.

For L2 let $x \neq y$, $x, y \notin X$ and $u \in \mathrm{Tm}^\alpha(X \cup \{y\}), v \in \mathrm{Tm}^\alpha(X)$, we have $1_{X\cup\{y\}}[x, u] \in X \cup \{x, y\} \to \mathrm{Tm}^\alpha(X \cup \{y\})$ and $1_X[y, v] \in X \cup \{y\} \to \mathrm{Tm}^\alpha(X)$. By 3. we have that $(\!|1_X[y, v]|\!) \circ (\!|1_{X\cup\{y\}}[x, u]|\!) = (\!|(\!|1_X[y, v]|\!) \circ 1_{X\cup\{y\}}[x, u]|\!)$. A simple case analysis on $z \in X \cup \{x, y\}$ using L1 shows that:

$$(\!|1_X[y, v]|\!) \circ 1_{X\cup\{y\}}[x, u])(z) = (1_{X\cup\{x\}}[x, (\!|1_X[y, v]|\!)(u)] \circ 1_{X\cup\{x\}}[y, v])(z)$$

Indeed, all algebraic properties of substitution (substitution laws) are derivable from prop 6 using only case analysis over variables.

To prove the proposition we observe that 2. follows directly from the definition of substitution. To verify 1. we have to show a more general lemma.

Lemma 7
Let $f \in X \to Y$ be an injective function (hence it is also a partial bijection and $f \in X \to \mathrm{Tm}^\alpha(Y)$). We have

$$(\!|f|\!)(t) \equiv^f_\alpha t$$

Proof
Induction over $t \in \mathrm{Tm}(X)$. I.e. consider $t = \lambda x.t'$ we have

$$(\!|f|\!)(\lambda x.t') = \lambda z.(\!|f[x = z]|\!)(t')$$

where $z = \mathrm{Fresh}(X)$. Since $x \notin X$ we have that $f[x = z] = f(x = z)$ and hence by ind.hyp. $(\!|f[x = z]|\!)(t') \equiv^{f(x=z)}_\alpha t'$ and therefore $\lambda x.t' \equiv^f_\alpha \lambda z.(\!|f[x = z]|\!)(t')$. □

Proof of proposition 6

1. By Lemma 7 by setting $f = \eta_X$.
2. Immediate from the definition of $(\!|f|\!)$.
3. We show

   $$((\!|f|\!) \circ (\!|g|\!))(t) \equiv_\alpha (\!|(\!|f|\!) \circ g|\!)(t)$$

   by induction over $t \in \mathrm{Tm}(X)$. As usual the interesting case is $t = \lambda x.t'$:

$$
\begin{array}{ll}
 & (\!|f|\!) \circ (\!|g|\!))(\lambda x.t') \\
= & \quad \{ \text{ defn of } (\!|-|\!), \text{ where } z_0 = \mathrm{Fresh}(Y) \ \} \\
 & (\!|f|\!)(\lambda z_0.(\!|g[x = z_0]|\!)(t)) \\
= & \quad \{ \text{ defn of } (\!|-|\!), \text{ where } z_1 = \mathrm{Fresh}(Z) \ \} \\
 & \lambda z_1.((\!|f[z_0 = z_1]|\!) \circ (\!|g[x = z_0]|\!))(t') \\
\equiv_\alpha & \quad \{ \text{ ind.hyp. } \} \\
 & \lambda z_1.(\!|(\!|f[z_0 = z_1]|\!) \circ g[x = z_0]|\!)(t') \\
\equiv_\alpha & \quad \{ \ (\!|f[z_0 = z_1]|\!) \circ g[x = z_0] \equiv_\alpha ((\!|f|\!) \circ g)[x = z_1], \text{ see below } \} \\
 & \lambda z_1.(\!|((\!|f|\!) \circ g)[x = z_1]|\!)(t') \\
= & \quad \{ \text{ defn of } (\!|-|\!) \ \} \\
 & (\!|(\!|f|\!) \circ g|\!)(\lambda x.t')
\end{array}
$$

We have to show $(\!|f[z_0 = z_1]|\!) \circ g[x = z_0](v) \equiv_\alpha ((\!|f|\!) \circ g)[x = z_1](v)$: If $v = x$ both sides evaluate to z_1, otherwise:

$$
\begin{array}{ll}
 & (\!|f[z_0 = z_1]|\!) \circ g[x = z_0])(v) \\
= & \quad \{ \ v \neq x \ \} \\
 & (\!|f[z_0 = z_1]|\!)(g(v)) \\
\equiv_\alpha & \quad \{ \ z_0 \notin Y, \text{ as for example 2. } \} \\
 & (\!|f|\!)(g(v)) \\
= & \quad \{ \ v \neq x \ \} \\
 & ((\!|f|\!) \circ g)[x = z_1](v)
\end{array}
$$

□

A monad gives rise to its Kleisli-category. In the case of Tm^α this is the category **Tm** of terms and substitutions: objects are $X \subset_{\mathrm{fin}} \mathcal{V}$ and morphisms are substitutions: $\mathbf{Tm}(X, Y) = X \to \mathrm{Tm}^\alpha(Y)$. The identity is given by the unit $\eta_X \in \mathbf{Tm}(X, X)$ and composition by lifting, i.e. $f \circ_{\mathbf{Tm}} g = (\!|f|\!) \circ g$. The categorical laws are a direct consequence of the monad laws (i.e. prop 6).

5 de Bruijn terms

To avoid the complications of α-conversion de Bruijn (de Bruijn, 1972) developed a representation of terms where variables are replaced by a number indicating the binding depth and λ-abstraction is an unary operation. E.g. the term $\lambda x.x(\lambda y.yx)$ becomes $\lambda 0(\lambda 01)$.

For any $n \in \mathrm{Nat}$ we define the set $\mathrm{Tm}^{\mathrm{db}}(n)$ of de Bruijn terms with at most n free Variables inductively by the following rules: [2]

$$\frac{i \in n}{i \in \mathrm{Tm}^{\mathrm{db}}(n)} \qquad \frac{t, u \in \mathrm{Tm}^{\mathrm{db}}(n)}{t\,u \in \mathrm{Tm}^{\mathrm{db}}(n)} \qquad \frac{t \in \mathrm{Tm}^{\mathrm{db}}(n+1)}{\lambda t \in \mathrm{Tm}^{\mathrm{db}}(n)}$$

The idea is that the elements of $\mathrm{Tm}^{\mathrm{db}}(n)$ can be considered as representations of the equivalence classes of $\mathrm{Tm}^\alpha(X)$ where $|X| \leq n$. We can make this precise, given $X \subset_{\mathrm{fin}} \mathcal{V}$ and an injection $\phi \in X \to n$, we assign to any $t \in \mathrm{Tm}(X)$ a de Bruijn

[2] Here we identify $n \in \mathrm{Nat}$ with the set $\{i \in \mathrm{Nat} \mid i < n\}$

term $t^\phi \in \mathrm{Tm}^{\mathrm{db}}(n)$ by

$$\begin{aligned} x^\phi &= \phi(x) \\ (tu)^\phi &= t^\phi u^\phi \\ (\lambda x.t) &= \lambda t^{\phi^{+x}} \end{aligned}$$

where

$$\phi^{+x}(y) = \begin{cases} 0 & \text{if } y = x \\ \phi(y) + 1 & \text{otherwise} \end{cases}$$

Note that ϕ^{+x} is an injection, if ϕ is.

We want to formally establish that the $-^\phi$ operation chooses a canonical representation for each α-equivalence class:

Proposition 8
$-^\phi$ is an injection between $\mathrm{Tm}^\alpha(X) \to \mathrm{Tm}^{\mathrm{db}}(n)$, i.e.

$$t \equiv_\alpha u \iff t^\phi = u^\phi$$

To show this we have establish the following generalisation:

Lemma 9
Given injections $\phi \in X \to n, \psi \in Y \to n$ we have

$$t \equiv_\alpha^R u \iff t^\phi = u^\psi$$

where R is the pullback of ϕ and ψ, i.e.

$$xRy \iff \phi(x) = \psi(y)$$

Proof
By induction of $t \in \mathrm{Tm}(X)$. As usual the interesting case is λ where we need the fact that if R is given as above then

$$vR(x,y)w \iff \phi^{+x}(v) = \psi^{+y}(w)$$

□

I leave it to the reader to show that $-^\phi$ preserves substitution, i.e. it maps substitutions on named terms as given here to substitution on de Bruijn terms, e.g. as defined in (Altenkirch & Reus, 1999).

6 Conclusions

Maybe the reader will have come to the conclusion that α-conversion isn't easy given the number of definitions, lemmas and propositions in this note. However, I would translate *easy* as: everything works out as expected, there is no creativity needed, no need to come up with unexpected technical lemmas. In this sense we can conclude *α-conversion is easy!*

The theory presented here offers a viable alternative to the use of de Bruijn terms or non-standard presentations of λ-terms in formal developments. All the propositions in this paper are provable constructively, hence the development could be formalized in a constructive metalanguage like Martin-Löf's Type Theory.

References

Altenkirch, Thorsten. (1993). A formalization of the strong normalization proof for System F in LEGO. *Pages 13 – 28 of:* M. Bezem, J.F. Groote (ed), *Typed lambda calculi and applications*. LNCS 664.

Altenkirch, Thorsten, & Reus, Bernhard. (1999). Monadic presentations of lambda terms using generalized inductive types. *Computer science logic*.

Barendregt, H.P. (1984). *The lambda calculus - its syntax and semantics (revised edition)*. Studies in Logic and the Foundations of Mathematics. North Holland.

Curry, H. B., & Feys, R. (1958). *Combinatory Logic*. Vol. I. North-Holland.

de Bruijn, N. G. (1972). Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. math.*, **34**(5), 381–392.

Fiore, Marcelo, Plotkin, Gordon, & Turi, Daniele. (1999). Abstract syntax and variable binding (extended abstract). *Pages 193–202 of: 14th annual symposium on logic in computer science*.

Gabbay, M. J., & Pitts, A. M. (1999). A new approach to abstract syntax involving binders. *Pages 214–224 of: 14th annual symposium on logic in computer science*.

Gordon, Andrew D., & Melham, Tom. (1996). Five axioms of alpha-conversion. *Pages 173–191 of:* von Wright, J., Grundy, J., & Harrison, J. (eds), *Proceedings of the 9th international conference on theorem proving in higher order logics (tphols'96)*. Turku, Finland: Springer-Verlag LNCS 1125.

Hindley, J. Roger, & Seldin, Jonathan P. (1986). *Introduction to combinators and λ-calculus*. Cambridge University Press.

Hofmann, Martin. (1999). Semantical analysis of higher-order abstract syntax. *14th annual symposium on logic in computer science*.

McKinna, James, & Pollack, Robert. (1993). Pure type systems formalized. M. Bezem, J.F. Groote (ed), *Typed lambda calculi and applications*. LNCS 664.