# Substitution Revisited[1]

Allen Stoughton

Computer Science Subject Group
School of Mathematical and Physical Sciences
University of Sussex
Falmer, Brighton BN1 9QH, England

**Abstract.** A definition of simultaneous substitution for the lambda calculus is presented that is easier to work with than standard single substitution because it is a definition by structural recursion, instead of recursion on the length of terms, and bound variables are always renamed. As a result, many proofs involving substitution are by structural induction, instead of induction on the length of terms, and are simpler than the corresponding standard proofs because of the reduction in the number of cases that must be considered. Furthermore, because of the uniform renaming of bound variables, identity substitutions normalize terms with respect to equivalence up to the renaming of bound variables ($\alpha$-congruence), allowing induction-free proofs of some theorems that ordinarily would be proved by induction on the relation of $\alpha$-congruence.

A series of results relating simultaneous substitution and $\alpha$-congruence are proved, and a simple proof of the "substitution lemma" of denotational semantics is given.

## 1 Introduction

Logics for languages in which variables can be bound generally involve substitution. Unrestricted, naive substitution leads to inconsistencies, since free variables may be captured in the process. Two ways of avoiding this problem are common. In the first, as in many presentations of the first order predicate calculus, substitution is performed in the naive way but is only allowed when free variables are not captured. In the second, as in the lambda calculus, unrestricted substitution is allowed but bound variables are renamed, as necessary, to avoid capturing. This paper focuses on unrestricted substitution for the untyped lambda calculus. Most of the definitions and results will, however, apply easily to other languages.

The standard definition of unrestricted substitution for the lambda calculus was given by Curry and Feys in [2] p. 94, where the notation $[M/x]N$ was used to denote the substitution of $M$ for the free occurrences of $x$ in $N$. (See also [6] p. 62 and [1] p. 578.) Unfortunately, proofs involving this definition of substitution are notoriously tedious. First, in cases when capturing would occur, $[M/x]\lambda y.N$ is defined to be $\lambda z.[M/x][z/y]N$, for a new variable $z$. Since $[z/y]N$ is not a subterm

[1] Appears in *Theoretical Computer Science*, 59:317–325, 1988.

of $\lambda y.N$, $[-/-]N$ is a definition by recursion on the length instead of the structure of $N$, and thus many proofs must be by induction on the length of $N$, and contain two applications of the inductive hypothesis for this subcase. Second, there are three subcases to the abstraction case of the definition, since bound variables are renamed only when necessary, and thus there are three subcases to consider in many proofs. Because bound variables must sometimes be renamed, it is necessary, in general, to work with the equivalence of terms up to the renaming of bound variables ($\alpha$-congruence) instead of identity. Complicating the definition by preserving identity whenever possible is thus questionable.

For examples of these complexities see [2] pp. 95–104 for the proofs of a series of basic theorems about substitution, and [6] pp. 161–166 for a proof of the "substitution lemma" of denotational semantics. (This proof of the substitution lemma is incorrectly claimed to be by structural induction; it is actually by induction on the length of terms.)

This paper gives a definition of simultaneous substitution that is by structural recursion, since bound variables are renamed in parallel with substitutions, and in which bound variables are always renamed. As a result, many proofs involving substitution are by structural induction, and are simpler than the corresponding standard proofs because of the reduction in the number of cases that must be considered. Furthermore, because of the uniform renaming of bound variables, identity substitutions normalize terms with respect to $\alpha$-congruence, allowing induction-free proofs of some theorems that ordinarily would be proved by induction on the relation of $\alpha$-congruence.

The idea of always renaming bound variables is fairly obvious, and is also used, e.g., in [4] p. 379. The key technique of using simultaneous substitution in order to give a definition by structural recursion also appears in [3] pp. 49–56, where substitution for the predicate calculus is defined. This paper's contribution is to develop a simple, unified theory, based upon these ideas.

Section 2 of the paper gives the definitions of simultaneous substitution and $\alpha$-congruence. Section 3 proves a series of results relating substitution and $\alpha$-congruence. Finally, section 4 gives a simple proof of the "substitution lemma" of denotational semantics. With one exception, the results of section 4 are independent from those of section 3.

## 2 Definitions

The application of a function $f$ to an argument $a$ is written $f\,a$. Function space formation, $D \to E$, associates to the right, and function application to the left. For $f\colon D \to E$, $d \in D$ and $e \in E$, the function $f[e/d]$ from $D$ to $E$ is defined by

$$f[e/d]\,d' = \begin{cases} e & \text{if } d' = d, \text{ and} \\ f\,d' & \text{otherwise.} \end{cases}$$

Two functions $f, g\colon D \to E$ are *equal over* a subset $X$ of $D$, written $f =^X g$, iff for all $x \in X$, $f\,x = g\,x$.

Let $V$ be a denumerable set of *variables*, and *choice* be a *choice function* for $V$, i.e., a function from $(\mathcal{P}V) - \{\emptyset\}$ to $V$ such that $choice\,X \in X$, for all nonempty $X \subseteq V$. The set of *terms* $T$ is least such that

$$\begin{aligned} x \in T &\text{ if } x \in V, \\ M\,N \in T &\text{ if } M \in T \text{ and } N \in T, \text{ and} \\ \lambda x.M \in T &\text{ if } x \in V \text{ and } M \in T. \end{aligned}$$

As usual, we use the lower case letters $u$, $v$, $w$, $x$, $y$ and $z$ and the upper case letters $M$ and $N$ to range over variables and terms, respectively.

Define the *free variable function* $\mathrm{FV}(-): T \to \mathcal{P}V$ by structural recursion:

$$\begin{aligned} \mathrm{FV}(x) &= \{x\}, \\ \mathrm{FV}(M\,N) &= \mathrm{FV}(M) \cup \mathrm{FV}(N), \text{ and} \\ \mathrm{FV}(\lambda x.M) &= \mathrm{FV}(M) - \{x\}. \end{aligned}$$

A variable $x$ is *free* in a term $M$ iff $x \in \mathrm{FV}(M)$.

The set of *substitutions* $S$ is $V \to T$, and we let $\sigma$ range over $S$. The *identity substitution* $\iota$ is defined by $\iota\, x = x$. Define $new: V \to T \to S \to ((\mathcal{P}V) - \{\emptyset\})$ by

$$new\, x\, M\, \sigma = \{\, y \mid \text{for all } z \in \mathrm{FV}(M) - \{x\},\ y \notin \mathrm{FV}(\sigma\, z)\,\},$$

so that $new\, x\, M\, \sigma$ contains all but a finite subset of $V$. The *simultaneous substitution* $M\,\sigma$ of $\sigma\, x$ for the free occurrences of $x$ in $M$, for all $x$, is defined by structural recursion:

$$\begin{aligned} x\,\sigma &= \sigma\, x, \\ (M\,N)\sigma &= (M\,\sigma)(N\,\sigma), \text{ and} \\ (\lambda x.M)\sigma &= \lambda y.(M\,\sigma[y/x]), \text{ where } y = choice(new\, x\, M\, \sigma). \end{aligned}$$

The *composition* $\sigma_2 \circ \sigma_1$ of substitutions $\sigma_1$ and $\sigma_2$ is defined by $(\sigma_2 \circ \sigma_1)x = (\sigma_1\, x)\sigma_2$. Note that $(\lambda x.M)\sigma$ is independent from $\sigma\, x$, a fact which, in addition to making intuitive sense, is necessary for the validity of much of section 3, e.g., theorems 3.2 and 3.5.

Let $=_\alpha$ be the relation of *$\alpha$-congruence*, i.e., the least equivalence relation over $T$ such that

($\mu$) $M\,N =_\alpha M'\,N'$ if $M =_\alpha M'$ and $N =_\alpha N'$; and

($\alpha$) $\lambda x.M =_\alpha \lambda y.N$ if either

(i) $x = y$ and $M =_\alpha N$, or

(ii) $y \notin \mathrm{FV}(M)$ and $M\,\iota[y/x] =_\alpha N$.

We say that $M$ and $N$ are *$\alpha$-congruent* iff $M =_\alpha N$. Substitutions $\sigma_1$ and $\sigma_2$ are *$\alpha$-congruent over* $X \subseteq V$, written $\sigma_1 =_\alpha^X \sigma_2$, iff $\sigma_1\, x =_\alpha \sigma_2\, x$, for all $x \in X$, and *$\alpha$-congruent*, written $\sigma_1 =_\alpha \sigma_2$, iff $\sigma_1 =_\alpha^V \sigma_2$. The hypothesis of condition ($\alpha$) is used frequently below, and is abbreviated by $M(x) =_\alpha N(y)$; it can be read as *$M$ and $N$ are $\alpha$-congruent up to the renaming of $x$ to $y$*. We will see below that $M(x) =_\alpha N(y)$ iff $N(y) =_\alpha M(x)$ (corollary 3.7).

## 3 Substitution and $\alpha$-Congruence

**Lemma 3.1** (i) *If $y \notin \mathrm{FV}(M)$ then $\lambda x.M =_\alpha \lambda y.(M\,\iota[y/x])$.*

(ii) *$x \in \mathrm{FV}(M\,\sigma)$ iff $x \in \mathrm{FV}(\sigma\, y)$, for some $y \in \mathrm{FV}(M)$.*

(iii) *If $M =_\alpha N$ then $\mathrm{FV}(M) = \mathrm{FV}(N)$.*

(iv) *If $M_1 =_\alpha M_2$ and $\sigma_1 =_\alpha^{\mathrm{FV}(M_1)-\{x\}} \sigma_2$ then $new\, x\, M_1\, \sigma_1 = new\, x\, M_2\, \sigma_2$.*

(v) *If $\sigma_1 =^{\mathrm{FV}(M)} \sigma_2$ then $M\,\sigma_1 = M\,\sigma_2$.*

(vi) *$M\,\iota =_\alpha M$*

(vii) *If $M(x) =_\alpha N(y)$ then $new\, x\, M\, \sigma = new\, y\, N\, \sigma$.*

(viii) *For $y \in new\, x\, M\, \sigma_1$, $new\, x\, M\, (\sigma_2 \circ \sigma_1) = new\, y\, (M\,\sigma_1[y/x])\, \sigma_2$.*

**Proof.** For (i), $M\,\iota[y/x] =_\alpha M\,\iota[y/x]$, by reflexivity, and the result follows from $(\alpha)$.

(ii) follows by structural induction over $M$.

(iii) is by induction on $=_\alpha$, i.e., define a relation $\approx \subseteq =_\alpha$ by $M \approx N$ iff $M =_\alpha N$ and $\mathrm{FV}(M) = \mathrm{FV}(N)$, and show that $\approx$ satisfies the defining conditions of $=_\alpha$. (ii) is used to show that $\approx$ satisfies $(\alpha)$.

(iv) is a consequence of (iii).

(v) is by structural induction on $M$, using (iv) for the abstraction case.

(vi) follows by structural induction on $M$, using (i) for the abstraction case.

(vii) follows easily from (iii).

We give a detailed proof of (viii), as an example. First, suppose $w \in new\,x\,M\,(\sigma_2 \circ \sigma_1)$. To show that $w \in new\,y\,(M\,\sigma_1[y/x])\,\sigma_2$, suppose $z \in \mathrm{FV}(M\,\sigma_1[y/x]) - \{y\}$; we must show that $w \notin \mathrm{FV}(\sigma_2\,z)$. Then $z \in \mathrm{FV}(\sigma_1[y/x]\,v)$, for some $v \in \mathrm{FV}(M)$, and, since $z \neq y$, it follows that $v \neq x$ and $z \in \mathrm{FV}(\sigma_1\,v)$. Since $w \in new\,x\,M\,(\sigma_2 \circ \sigma_1)$ and $v \in \mathrm{FV}(M) - \{x\}$, $w \notin \mathrm{FV}((\sigma_2 \circ \sigma_1)v) = \mathrm{FV}((\sigma_1\,v)\sigma_2)$, and thus $w \notin \mathrm{FV}(\sigma_2\,u)$, for all $u \in \mathrm{FV}(\sigma_1\,v)$. But $z \in \mathrm{FV}(\sigma_1\,v)$, and thus $w \notin \mathrm{FV}(\sigma_2\,z)$, as required. Second, suppose $w \in new\,y\,(M\,\sigma_1[y/x])\,\sigma_2$. To show that $w \in new\,x\,M\,(\sigma_2 \circ \sigma_1)$, suppose $z \in \mathrm{FV}(M) - \{x\}$. We must show that $w \notin \mathrm{FV}((\sigma_1\,z)\sigma_2)$, and this will follow from showing that $w \notin \mathrm{FV}(\sigma_2\,v)$, under the assumption that $v \in \mathrm{FV}(\sigma_1\,z)$. Then $v \in \mathrm{FV}(\sigma_1[y/x]\,z)$, since $z \neq x$, and thus $v \in \mathrm{FV}(M\,\sigma_1[y/x])$. Furthermore, $v \neq y$, since, by its definition, $y \notin \mathrm{FV}(\sigma_1\,z)$. But then $v \in \mathrm{FV}(M\,\sigma_1[y/x]) - \{y\}$, and, since $w \in new\,y\,(M\,\sigma_1[y/x])\,\sigma_2$, we can conclude that $w \notin \mathrm{FV}(\sigma_2\,v)$, as required. □

From (v) and (vi) of lemma 3.1, if $x \notin \mathrm{FV}(M)$ then $M\,\iota[N/x] =_\alpha M$.

The following *Syntactic Substitution Theorem* shows that performing the composition of two substitutions yields the same result as performing those substitutions sequentially.

**Theorem 3.2** $(M\,\sigma_1)\sigma_2 = M(\sigma_2 \circ \sigma_1)$

**Proof.** By structural induction on $M$. The variable and application cases are trivial. For an abstraction $\lambda x.M$,

$$\begin{aligned}((\lambda x.M)\sigma_1)\sigma_2 &= (\lambda y.(M\,\sigma_1[y/x]))\sigma_2 \\ &= \lambda y'.((M\,\sigma_1[y/x])\sigma_2[y'/y]) \\ &= \lambda y'.(M\,(\sigma_2[y'/y] \circ (\sigma_1[y/x]))) \qquad \text{(induction)},\end{aligned}$$

where $y = choice(new\,x\,M\,\sigma_1)$ and $y' = choice(new\,y\,(M\,\sigma_1[y/x])\,\sigma_2)$, and

$$(\lambda x.M)(\sigma_2 \circ \sigma_1) = \lambda z.(M(\sigma_2 \circ \sigma_1)[z/x]),$$

where $z = choice(new\,x\,M\,(\sigma_2 \circ \sigma_1))$. But $z = y'$, by lemma 3.1 (viii), and

$$M(\sigma_2[z/y] \circ (\sigma_1[y/x])) = M\,(\sigma_2 \circ \sigma_1)[z/x]$$

follows by two applications of lemma 3.1 (v). □

**Corollary 3.3** *If* $y \notin \mathrm{FV}(M)$ *then* $(M\,\iota[y/x])\sigma[N/y] = M\,\sigma[N/x]$.

**Proof.** Follows easily from theorem 3.2 and lemma 3.1 (v). □

**Corollary 3.4** (i) $\iota \circ \sigma =_\alpha \sigma = \sigma \circ \iota$
(ii) $\sigma_3 \circ (\sigma_2 \circ \sigma_1) = (\sigma_3 \circ \sigma_2) \circ \sigma_1$

**Proof.** (i) is by lemma 3.1 (vi), and (ii) follows easily from theorem 3.2. □

The following theorem, the basis of the remainder of the section, is remarkable: applying a substitution to each of two $\alpha$-congruent terms yields equal—not just $\alpha$-congruent—results! An immediate corollary is that identity substitutions normalize terms with respect to $\alpha$-congruence, a fact which allows induction-free proofs of some theorems (like corollaries 3.10 and 4.5) that ordinarily would be proved by induction on $=_\alpha$.

**Theorem 3.5** *If* $M =_\alpha N$ *then* $M\,\sigma = N\,\sigma$.

**Proof.** By induction on $=_\alpha$, i.e., define a relation $\approx \subseteq =_\alpha$ by $M \approx N$ iff $M =_\alpha N$ and $M\,\sigma = N\,\sigma$, for all $\sigma$, and show that $\approx$ satisfies the defining conditions of $=_\alpha$. Obviously $\approx$ is an equivalence relation and satisfies $(\mu)$. For $(\alpha)$, suppose that either
(i) $x = y$ and $M \approx N$, or
(ii) $y \notin \mathrm{FV}(M)$ and $M\,\iota[y/x] \approx N$.
We must show that $(\lambda x.M)\sigma = (\lambda y.N)\sigma$, for all $\sigma$. By lemma 3.1 (vii), $new\,x\,M\,\sigma = new\,y\,N\,\sigma$, and thus

$$(\lambda x.M)\sigma = \lambda z.(M\,\sigma[z/x])$$

and

$$(\lambda y.N)\sigma = \lambda z.(N\,\sigma[z/y]),$$

for $z = choice(new\,x\,M\,\sigma)$. If (i) then $M\,\sigma[z/x] = N\,\sigma[z/x] = N\,\sigma[z/y]$, by induction. Alternatively, if (ii) then $M\,\sigma[z/x] = (M\,\iota[y/x])\sigma[z/y] = N\,\sigma[z/y]$, by corollary 3.3 and induction. □

**Corollary 3.6** (i) $M =_\alpha N$ *iff* $M\,\iota = N\,\iota$
(ii) $\sigma_1 =_\alpha^X \sigma_2$ *iff* $\iota \circ \sigma_1 =^X \iota \circ \sigma_2$

**Proof.** (i) follows from theorem 3.5 and lemma 3.1 (vi), and (ii) is immediate from (i). □

Surprisingly, we can only now prove the following simply stated result.

**Corollary 3.7** $M(x) =_\alpha N(y)$ *iff* $N(y) =_\alpha M(x)$

**Proof.** Follows easily from lemma 3.1 (iii) and (vi), corollary 3.3 and theorem 3.5. □

The following corollary shows that substitution and $\alpha$-congruence are compatible.

**Corollary 3.8** *If* $M_1 =_\alpha M_2$ *and* $\sigma_1 =_\alpha^{\mathrm{FV}(M_1)} \sigma_2$ *then* $M_1\,\sigma_1 =_\alpha M_2\,\sigma_2$.

**Proof.** By corollary 3.6 (ii), $\iota \circ \sigma_1 =^{\mathrm{FV}(M_1)} \iota \circ \sigma_2$, and thus

$$M_1\,\sigma_1 =_\alpha (M_1\,\sigma_1)\iota = M_1(\iota \circ \sigma_1) = M_1(\iota \circ \sigma_2) = M_2(\iota \circ \sigma_2) = (M_2\,\sigma_2)\iota =_\alpha M_2\,\sigma_2,$$

by lemma 3.1 (vi), theorem 3.2, lemma 3.1 (v) and theorem 3.5. □

The following is a companion result to corollary 3.3. Unfortunately, it cannot be strengthened from $=_\alpha$ to $=$.

**Corollary 3.9** *If $y \in new\, x\, M\, \sigma$ then $(M\,\sigma[y/x])\iota[N/y] =_\alpha M\,\sigma[N/x]$.*

**Proof.** Follows easily from theorem 3.2 and corollary 3.8. □

Now we are able to characterize the structure of $\alpha$-congruence.

**Corollary 3.10** *If $M =_\alpha N$ then one of the following conditions holds:*
(i) *$M$ and $N$ are equal variables;*
(ii) *$M$ and $N$ are applications $M_1\, M_2$ and $N_1\, N_2$, respectively, and $M_i =_\alpha N_i$, for $i = 1, 2$;*
(iii) *$M$ and $N$ are abstractions $\lambda x.M'$ and $\lambda y.N'$, respectively, and $M'(x) =_\alpha N'(y)$.*

**Proof.** If $M =_\alpha N$ then $M\,\iota = N\,\iota$, by corollary 3.6 (i). There are three cases to consider.

(i) If $M$ is a variable $x$ then $x = x\,\iota = N\,\iota$, and thus $N = x$.

(ii) If $M$ is an application $M_1\, M_2$ then $(M_1\,\iota)(M_2\,\iota) = N\,\iota$, and thus $N$ is an application $N_1\, N_2$, and $M_i\,\iota = N_i\,\iota$, for $i = 1, 2$. But then $M_i =_\alpha N_i$, for $i = 1, 2$, by corollary 3.6 (i).

(iii) If $M$ is an abstraction $\lambda x.M'$ then $\lambda z.(M'\,\iota[z/x]) = N\,\iota$, for $z = choice(new\, x\, M'\,\iota)$, and thus $N$ is an abstraction $\lambda y.N'$, and $(\lambda y.N')\iota = \lambda z.(N'\,\iota[z/y])$. Since $M'\,\iota[z/x] = N'\,\iota[z/y]$,

$$M'\,\iota[y/x] =_\alpha (M'\,\iota[z/x])\iota[y/z] = (N'\,\iota[z/y])\iota[y/z] =_\alpha N'\,\iota[y/y] = N'\,\iota =_\alpha N',$$

by corollary 3.9. If $x = y$ then $M' =_\alpha M'\,\iota = M'\,\iota[y/x] =_\alpha N'$. Alternatively, if $x \neq y$ then $y \notin \mathrm{FV}(M')$, since $y \notin \mathrm{FV}(\lambda y.N') = \mathrm{FV}(\lambda x.M')$. In either case, $M'(x) =_\alpha N'(y)$, as required. □

Our final corollary shows that, up to $\alpha$-congruence, any element of $new\, x\, M\, \sigma$ may be chosen as the bound variable of $(\lambda x.M)\sigma$.

**Corollary 3.11** *If $y \in new\, x\, M\, \sigma$ then $(\lambda x.M)\sigma =_\alpha \lambda y.(M\,\sigma[y/x])$.*

**Proof.** Follows from corollary 3.9. □

## 4 Substitution and Denotational Semantics

This section consists of a proof of the "substitution lemma" of denotational semantics. Familiarity with some standard definitions and results about complete partial orders (cpo's) and continuous functions, which can be found, e.g., in [5], is assumed. Our denotational semantics is taken from [7], with the exception that cpo's instead of complete lattices are used.

Let the cpo $E$ of *expression values* be a nontrivial solution to the isomorphism equation $E \cong E \to E$, and $in\colon (E \to E) \to E$ and $out\colon E \to (E \to E)$ be continuous functions such that $out \circ in = id_{E \to E}$ and $in \circ out = id_E$. Let the cpo $U$ of *environments* be $V \to E$, ordered componentwise; we use $\rho$ to range over $U$. Define a denotational semantics $\mathcal{E}\colon T \to U \to E$ by structural recursion:

$$\begin{aligned} \mathcal{E}[\![x]\!] &= \lambda\rho.\rho\, x, \\ \mathcal{E}[\![M\, N]\!] &= \lambda\rho.(out\, \mathcal{E}[\![M]\!]\rho)\mathcal{E}[\![N]\!]\rho, \text{ and} \\ \mathcal{E}[\![\lambda x.M]\!] &= \lambda\rho.in\, \lambda e.\mathcal{E}[\![M]\!]\rho[e/x]. \end{aligned}$$

The *composition* $\rho \circ \sigma$ of an environment $\rho$ and a substitution $\sigma$ is the environment defined by $(\rho \circ \sigma)x = \mathcal{E}[\![\sigma\, x]\!]\rho$.

**Lemma 4.1** *If* $\rho_1 =^{\mathrm{FV}(M)} \rho_2$ *then* $\mathcal{E}[\![M]\!]\rho_1 = \mathcal{E}[\![M]\!]\rho_2$.

**Proof.** An easy structural induction over $M$. □

We can now state and prove the simultaneous substitution form of the "substitution lemma". A specialization to single substitution follows as a corollary.

**Theorem 4.2** $\mathcal{E}[\![M\,\sigma]\!]\rho = \mathcal{E}[\![M]\!](\rho \circ \sigma)$

**Proof.** By induction on the structure of $M$. The variable and application cases are obvious. For an abstraction $\lambda x.M$,

$$\begin{aligned}\mathcal{E}[\![(\lambda x.M)\sigma]\!]\rho &= \mathcal{E}[\![\lambda y.(M\,\sigma[y/x])]\!]\rho \\ &= in\,\lambda e.\mathcal{E}[\![M\,\sigma[y/x]]\!]\rho[e/y] \\ &= in\,\lambda e.\mathcal{E}[\![M]\!](\rho[e/y] \circ \sigma[y/x]) \qquad \text{(induction)},\end{aligned}$$

where $y = choice(new\,x\,M\,\sigma)$, and

$$\mathcal{E}[\![\lambda x.M]\!](\rho \circ \sigma) = in\,\lambda e.\mathcal{E}[\![M]\!](\rho \circ \sigma)[e/x].$$

Thus, by lemma 4.1, it is sufficient to show that

$$(\rho[e/y] \circ \sigma[y/x])z = ((\rho \circ \sigma)[e/x])z,$$

for all $z \in \mathrm{FV}(M)$. If $z = x$ then both sides of this equation are $e$. Alternatively, if $z \neq x$ then

$$\begin{aligned}(\rho[e/y] \circ \sigma[y/x])z &= \mathcal{E}[\![\sigma\,z]\!]\rho[e/y] \\ &= \mathcal{E}[\![\sigma\,z]\!]\rho \qquad \text{(lemma 4.1)} \\ &= ((\rho \circ \sigma)[e/x])z,\end{aligned}$$

since, by its definition, $y \notin \mathrm{FV}(\sigma\,z)$. □

Theorem 4.2 is easily seen to be the semantic analogue of theorem 3.2, the Syntactic Substitution Theorem.

**Corollary 4.3** $\mathcal{E}[\![M\,\iota[N/x]]\!]\rho = \mathcal{E}[\![M]\!]\rho[\mathcal{E}[\![N]\!]\rho/x]$

**Proof.** Immediate from theorem 4.2. □

**Corollary 4.4** (i) $\rho \circ \iota = \rho$
(ii) $\rho \circ (\sigma_2 \circ \sigma_1) = (\rho \circ \sigma_2) \circ \sigma_1$

**Proof.** (i) is obvious, and (ii) is a consequence of theorem 4.2. □

All of the proofs presented so far in this section are completely independent from the results of section 3. A direct proof of the following, final corollary is also possible, by induction on $=_\alpha$; we prefer, however, to give an induction-free proof, exploiting the normalizing effect of substitution.

**Corollary 4.5** *If* $M =_\alpha N$ *then* $\mathcal{E}[\![M]\!] = \mathcal{E}[\![N]\!]$.

**Proof.** If $M =_\alpha N$ then $M\,\iota = N\,\iota$, by corollary 3.6, and thus

$$\mathcal{E}[\![M]\!]\rho = \mathcal{E}[\![M]\!](\rho \circ \iota) = \mathcal{E}[\![M\,\iota]\!]\rho = \mathcal{E}[\![N\,\iota]\!]\rho = \mathcal{E}[\![N]\!](\rho \circ \iota) = \mathcal{E}[\![N]\!]\rho,$$

for all $\rho$, showing that $\mathcal{E}[\![M]\!] = \mathcal{E}[\![N]\!]$. □

**Acknowledgements**

Marek Bednarczyk and Matthew Hennessy read a draft of this paper, and showed that several of its theorems could be strengthened from $\alpha$-congruence to equality. In particular, they proved that identity substitutions normalize terms with respect to $\alpha$-congruence, allowing induction-free proofs of two theorems that I originally proved by induction over the relation of $\alpha$-congruence.

I was financially supported by a University of Edinburgh studentship and a research fellowship from the Science and Engineering Research Council of Great Britain.

**References**

[1] H. Barendregt, *The Lambda Calculus: Its Syntax and Semantics* (North-Holland, Amsterdam, 1984).

[2] H. Curry and R. Feys, *Combinatory Logic*, Vol. I (North-Holland, Amsterdam, 1958).

[3] H. Ebbinghaus, J. Flum and W. Thomas, *Mathematical Logic* (Springer-Verlag, Berlin, 1984).

[4] G. Revesz, Axioms for the theory of lambda-conversion, *SIAM J. Comput.* **14** (2) (1985).

[5] M. Smyth and G. Plotkin, The category-theoretic solution of recursive domain equations, *SIAM J. Comput.* **11** (4) (1982).

[6] J. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory* (MIT Press, Cambridge, 1977).

[7] C. Wadsworth, The relation between computational and denotational properties for Scott's $D_\infty$-models of the lambda-calculus, *SIAM J. Comput.* **5** (3) (1976).