# A SIMPLER VERSION OF BASIC SIMPLE TYPE THEORY

ÁLVARO TASISTRO

Abstract. We give detailed proofs of the substitution lemmas for parallel reduction and simple type assignment in lambda calculus using its original syntax, i.e. named variables with no distinction between symbols to be used bound or free, and working directly on terms and not on $\alpha$ equivalence classes thereof. This is done using Stoughton's definition of substitution, which proves to be the right one for developing the theory of lambda calculus in the way stated.

## 1. Introduction

Our main purpose is to give fully detailed, formal proofs of the main meta-theoretical results of lambda calculus and its system of (simple) type assignment (a.k.a. simply typed lambda calculus à la Curry) using the original syntax of calculus, i.e. named variables with no distinction between symbols to be used bound or free, and working directly on terms and not on $\alpha$ equivalence classes thereof. This is done in [3, Basic Simple Type Theory] and its close relative [4], but with some limitations. Many proofs are not shown in detail, but referred back to [2, Curry and Feys] and the proofs shown are complicated and painful to formalize We want to investigate Stoughton's formulation of substitution and $\alpha$ conversion [5] as an alternative to the traditional formulation in order to assess how much it can simplify a fully formal development of the kind stated at the beginning. We intend the work also as a basis for an alternative method of formalisation of this theory in proof assistants like e.g. the ones based on type theory. So the development to be exposed should be regarded also as a blueprint of such a formalisation.

The paper can actually be described as an essay on simple proofs of the syntactic substitution lemmas of pure and simply typed $\lambda$ calculus with the traditional syntax and no simplifying conventions.

We begin the work in section 2 revisiting Stoughton's theory of substitution and $\alpha$ conversion which, by the way, includes a remarkable simple proof of the corresponding substitution lemma. This amounts to a (slight, but we hope, useful) reformulation of [5]. Next we prove in section 3 the substitution lemma for the parallel (or minimal complete development) reduction, which is at the basis of Tait–Martin-Löf's method of proof of confluence of $\beta$ reduction. In section 4 we do the same with the substitution lemma for the system of simple type assignment. The conclusions are exposed in section 5.

## 2. Syntax and Computation

2.1. **Preliminaries.** Application of (partial) function $f$ with domain $dom\, f$ to argument $a \in dom\, f$ is written $f\, a$. The usual overriding operation is written

*Date*: December 12, 2011.

$\_[\_ \mapsto \_]$, i.e.

$$dom\,(f[a \mapsto v]) = dom\,f \cup \{a\}$$
$$(f[a \mapsto v])\,x = \begin{cases} v & \text{if } x = a \\ f\,x & \text{otherwise} \end{cases} \quad \text{for } x \in dom\,(f[a \mapsto v])$$

2.2. **Terms.** The set $\mathbb{T}$ of terms is defined as usual from a denumerable set $\mathbb{V}$ of variables $x$ by the following (abstract) syntax:

$$M :- x|MN|\lambda xM$$

In concrete syntax we use the usual convention according to which application binds tighter than abstraction. Also define as usual the set $FV\,M$ of free variables of term $M$, by structural recursion:

$$\begin{array}{lcl} FV\,x & = & \{x\} \\ FV\,(MN) & = & FV\,M \cup FV\,N \\ FV\,\lambda xM & = & FV\,M - \{x\} \end{array}$$

2.3. **Substitutions.** Substitutions $\sigma$ are total functions $\mathbb{V} \to \mathbb{T}$, identity almost everywhere. We use $\iota$ for the identity substitution.

The effect $M\sigma$ of a substitution $\sigma$ on a term $M$ is now defined as in [5] just by structural recursion on terms, as follows:

$$\begin{cases} x\sigma & = & \sigma\,x \\ (MN)\sigma & = & M\sigma\ N\sigma \\ (\lambda xM)\sigma & = & \lambda yM(\sigma[x \mapsto y]) \text{ where } y = choice\,(\mathbb{V} - \bigcup_{z \in FV\,\lambda xM} FV\,(\sigma\,z)) \end{cases}$$

It is the simultaneous substitution of $\sigma\,x$ for the free occurrences of $x$ in $M$, for all variables $x$. When writing substitutions on terms we will use the convention that substitution binds stronger than application. The definition above is so given that substitution on an abstraction always performs renaming of the bound variable so that no capture occurs. This is ensured by choosing the new bound variable $y$ not to coincide with any free variable that is to appear in the body of the abstraction by effect of the substitution. The function *choice* above should be implemented in such a way that $choice\,X \in X$ for $X \neq \emptyset$. The point that (the choice of) $y$ can be expressed concisely as a function of the free variables of the abstraction and of those of the images thereof in the substitution is a fundamental insight in [5]. We write $\mathbb{W}^{\sigma}_{N}$ for $\bigcup_{z \in FV\,N} FV\,(\sigma\,z)$. Notice that $\mathbb{W}^{\sigma}_{N}$ is finite and thus $\mathbb{V} - \mathbb{W}^{\sigma}_{\lambda xM}$ is always non-empty. This latter set is called $new\,x\,M\,\sigma$ in [5]. $\mathbb{W}^{\sigma}_{\lambda xM}$ is the set of variables excluded for the choice of new bound variable in a renaming like in the third equation of the definition above. Also note that the change of bound variable is recorded as an overriding to the original substitution, which allows for the use of structural recursion.

The traditional definition of (single) substitution [2, 3, 4] is not by structural recursion like the present one. It rather goes by recursion on the size of terms although, actually, its justification must run simultaneously with a proof that substitution of a variable for another variable does not modify the size of the term, a fact which is never mentioned in any of the cited works. The question may be inessential for the usual mathematical textbook presentation, but has dramatic consequences as to the complete formalisation of the theory in a system like e.g. type theory as implemented in the available proof assistants.

Now we develop a theory of substitutions and $\alpha$ conversion similar, but not identical, to the one in section 3 of [5]. We shall point out the coincidences and

differences between the two approaches as we progress in the development. The theory covers (the analogues to) the results on single substitution and $\alpha$ conversion exposed in [3, 4]. In comparison with these, a consequence of the greater simplicity in the definition of substitution on a term is of course a corresponding simplification of the methods of proof which, in our case, are reduced to structural induction on terms. We start with definitions:

The *composition* $\sigma_2 \circ \sigma_1$ of two substitutions $\sigma_1$ and $\sigma_2$ is defined by the stipulation

$$(\sigma_2 \circ \sigma_1)\, x = (\sigma_1\, x)\sigma_2$$

Two substitutions $\sigma_1$ and $\sigma_2$ *coincide at* $M$, which we write $\sigma_1 =_M \sigma_2$, if they coincide at the free variables of $M$.

The first result and its method of proof are slightly more general than the corresponding ones in [5] so we shall spell them out in detail:

**Lemma 1.** $\sigma_1 =_M \sigma_2 \Leftrightarrow M\sigma_1{=}M\sigma_2$.

*Proof.* By structural induction on $M$. For the case of a variable $x$ we have $\sigma_1 =_x \sigma_2 \Leftrightarrow \sigma_1\, x{=}\sigma_2\, x \Leftrightarrow x\sigma_1{=}x\sigma_2$. For the case of an application $MN$, $\sigma_1 =_{MN} \sigma_2 \Leftrightarrow$ both $\sigma_1 =_M \sigma_2$ and $\sigma_1 =_N \sigma_2 \Leftrightarrow$ (induction) $M\sigma_1{=}M\sigma_2$ and $N\sigma_1{=}N\sigma_2 \Leftrightarrow M\sigma_1 N\sigma_1{=}M\sigma_2 N\sigma_2 \Leftrightarrow (MN)\sigma_1{=}(MN)\sigma_2$. For the case of an abstraction, we first prove the direction from left to right as follows: Assume $\sigma_1 =_{\lambda xM} \sigma_2$. We have to prove $(\lambda xM)\sigma_1{=}(\lambda xM)\sigma_2$. The left hand side is $\lambda y M\sigma_1[x \mapsto y]$ and the right hand side is $\lambda y' M\sigma_2[x \mapsto y']$ where $y = \mathit{choice}\,(\mathbb{V} - \mathbb{W}^{\sigma_1}_{\lambda xM})$ and $y' = \mathit{choice}\,(\mathbb{V} - \mathbb{W}^{\sigma_2}_{\lambda xM})$. But *choice* is in these two cases applied to the same set, as we readily see since from $\sigma_1 =_{\lambda xM} \sigma_2$ it follows for each $z \in FV(\lambda xM)$ that $\sigma_1\, z{=}\sigma_2\, z$, and further so their sets of free variables. Therefore $y = y'$ and the substitutions $\sigma_1[x \mapsto y]$ and $\sigma_2[x \mapsto y']$ coincide at $M$, since they already did so at $FVM - \{x\}$ and now we see they coincide at $x$ too. Now this gives by induction $M\sigma_1[x \mapsto y]{=}M\sigma_2[x \mapsto y']$ and, together with $y = y'$, the desired conclusion. For the direction from right to left, we start at $(\lambda xM)\sigma_1{=}(\lambda xM)\sigma_2$. The left hand side is $\lambda y M\sigma_1[x \mapsto y]$ and the right hand side is $\lambda y' M\sigma_2[x \mapsto y']$ for appropriately chosen $y$ and $y'$. But if those two abstractions are the same, it means $y = y'$ and $M\sigma_1[x \mapsto y] = M\sigma_2[x \mapsto y]$. This implies by induction that $\sigma_1[x \mapsto y] =_M \sigma_2[x \mapsto y]$ which in turn yields $\sigma_1 =_{\lambda xM} \sigma_2$. □

The next is stated for future use:

**Lemma 2.** (1) *If* $y \notin \mathbb{W}^{\sigma_1}_{\lambda xM}$ *then* $\mathbb{W}^{\sigma_2\circ\sigma_1}_{\lambda xM}{=}\mathbb{W}^{\sigma_2}_{\lambda yM\sigma_1[x\mapsto y]}$.
(2) $x \in FV(M\sigma)$ *iff* $x \in FV(\sigma\, y)$ *for some* $y \in FVM$.
(3) *If* $y \notin FVM$ *then* $FV\lambda xM{=}FV\lambda yM\iota[x \mapsto y]$.
(4) *If* $y \notin FVM$ *then* $\mathbb{W}^{\sigma}_{\lambda xM}{=}\mathbb{W}^{\sigma}_{\lambda yM\iota[x\mapsto y]}$.

*Proof.* For (1) see detailed proof of analogous statement in [5, Lemma 3.1 (viii)], consisting of not long but boring calculations. (2), given also in [5], is by structural induction on $M$. (3) is an easy calculation from (2) and (4) an immediate consequence of (3). □

The next result is called *Syntactic Substitution Theorem* in [5]. It establishes that, as it was to be expected, the effect of the composition of two substitutions is that of their sequential application:

**Theorem 1.** $M(\sigma_2 \circ \sigma_1){=}(M\sigma_1)\sigma_2$.

*Proof.* The theorem is of course proven in [5]. It proceeds just by structural induction on $M$. We include the only interesting case, namely that of an abstraction, mainly for completeness but also because we give a detailed account of the argument exposed in [5].

So, consider $(\lambda xM)(\sigma_2 \circ \sigma_1)$. By definition of the effect of a substitution, this is $\lambda yM(\sigma_2 \circ \sigma_1)[x \mapsto y]$ (1) where $y = \mathit{choice}(\mathbb{V} - \mathbb{W}^{\sigma_2\circ\sigma_1}_{\lambda xM})$. On the other hand, $((\lambda xM)\sigma_1)\sigma_2{=}(\lambda y'M\sigma_1[x \mapsto y'])\sigma_2{=}\ \lambda y''(M\sigma_1[x \mapsto y'])\sigma_2[y' \mapsto y'']$ (2) where $y' = \mathit{choice}(\mathbb{V} - \mathbb{W}^{\sigma_1}_{\lambda xM})$ and $y'' = \mathit{choice}(\mathbb{V} - \mathbb{W}^{\sigma_2}_{\lambda y'M\sigma_1[x\mapsto y']})$. Now we observe that we can apply Lemma 2(2) to $y'$, $\mathbb{W}^{\sigma_2\circ\sigma_1}_{\lambda xM}$ and $\mathbb{W}^{\sigma_2}_{\lambda y'M\sigma_1[x\mapsto y']}$, thus yielding $y = y''$. Therefore we write (2) as $\lambda y(M\sigma_1[x \mapsto y'])\sigma_2[y' \mapsto y]$ which, by induction, is $\lambda yM(\sigma_2[y' \mapsto y] \circ \sigma_1[x \mapsto y'])$ (3). Now we show that $(\sigma_2[y' \mapsto y] \circ \sigma_1[x \mapsto y']) =_M (\sigma_2 \circ \sigma_1)[x \mapsto y]$, which, by Lemma 1, implies the equality of (1) and (3) and thereby the desired result. Indeed, take $z \in FVM$. If $z = x$ then both substitutions yield $y$. On the other hand, if $z \neq x$, the first substitution yields $(\sigma_1\, z)\sigma_2$ and the second one $(\sigma_1\, z)\sigma_2[y' \mapsto y]$. Now, since $y' \notin FV(\sigma_1\, z)$ for $z \in FVM - \{x\}$, we have $\sigma_2[y' \mapsto y] =_{\sigma_1\, z} \sigma_2$, thus establishing the claim. □

The next result exposes a useful distributive law of composition over overriding:

**Lemma 3.** $\sigma_2 \circ \sigma_1[x \mapsto N]{=}(\sigma_2 \circ \sigma_1)[x \mapsto N\sigma_2]$.

*Proof.* Just check the definitions. □

**Corollary 1.** (1) $(M\iota[x \mapsto N])\sigma{=}M\sigma[x \mapsto N\sigma]$.
(2) *If* $y \notin FVM$ *then* $(M\iota[x \mapsto y])\sigma[y \mapsto N]{=}M\sigma[x \mapsto N]$.

*Proof.* Immediate. For (2) use Lemma 1. It is Corollary 3.3 of [5], though the proof is different. □

2.4. **Alpha conversion.** A *change of bound variable* is the "contraction"

$$\lambda xM \rhd_\alpha\ \ \lambda yM\iota[x \mapsto y]$$

where $y \neq x$ and $y \notin FVM$

The compatible (with term constuctors), reflexive and transitive closure of $\rhd_\alpha$ is $\equiv_\alpha$. This is symmetric, as we shall presently prove, and it is thus a congruence over terms, which we call $\alpha$ *conversion.*

This definition of $\alpha$ conversion departs from the one in [5]. It is rather the one in [3] or [4]. We prefer it because it generates $\equiv_\alpha$ from the basic action of change of bound variable.

A first result connecting $\equiv_\alpha$ and substitution is the following (cf. Lemma 3.1(vi) in [5].)

**Lemma 4.** $M\iota{\equiv_\alpha}M$

*Proof.* By induction on $M$. Of course the case to check is that of an abstraction: $(\lambda xM)\iota{=}\ \lambda yM\iota[x \mapsto y]$ (1) where $y \notin FV(\iota\, z)$ for $z \in FVM - \{x\}$. Now either $y = x$ or $y \neq x$. In the first case, the right hand side of (1) becomes $\lambda yM\iota$ where, by induction hypothesis, $M\iota{\equiv_\alpha}M$. In the second case, we have in addition

$y \notin FVM - \{x\}$ and thus further $y \notin FVM$. So (1) becomes exactly a change of bound variable. In any case then, the two terms in (1) are $\alpha$ convertible. □

Now we can show the symmetry of $\equiv_\alpha$.

**Lemma 5.** *$\equiv_\alpha$is symmetric.*

*Proof.* $\triangleright_\alpha$ is not symmetric. To go back from $\lambda yM\iota[x \mapsto y]$ to $\lambda xM$ we need more than one change of bound variable. But we can of course show $\lambda yM\iota[x \mapsto y] \equiv_\alpha \lambda xM$ and that is enough to prove the symmetry of $\equiv_\alpha$.

So start with $\lambda yM\iota[x \mapsto y]$ and change $y$ to $x$. Observe first that $x \notin FV(M\iota[x \mapsto y])$, since $y \neq x$. Moreover, $(M\iota[x \mapsto y])\iota[y \mapsto x] = M\iota$, which follows by applying Corollary 1(2) using the condition $y \notin FVM$. In turn, $M\iota \equiv_\alpha$M and this carries over the term constructor $\lambda x$, yielding the result. □

We now extend $\equiv_\alpha$ to substitutions, in the obvious way. We then write $\sigma \equiv_\alpha \sigma'$ for $\alpha$ convertible substitutions and $\sigma \equiv_{\alpha M} \sigma'$ for substitutions $\sigma$ and $\sigma'$ which are $\alpha$ convertible at the free variables of $M$.

The following is proven in [5]:

**Theorem 2.** (1) $\iota \circ \sigma \equiv_\alpha \sigma = \sigma \circ \iota$.
(2) $\sigma_3 \circ (\sigma_2 \circ \sigma_1) = (\sigma_3 \circ \sigma_2) \circ \sigma_1$

An easy result to be used later is the following:

**Lemma 6.** *If $M \equiv_\alpha N$ then $FVM = FVN$.*

*Proof.* The result is proven in [5] by induction on $\equiv_\alpha$. However, we have given a different definition of this relation, so the proof will run differently. The only interesting case is the one of a change of bound variable ($\alpha$ contraction.). But this is lemma 2(3) and has already been considered. □

The following is obvious.

**Corollary 2.** *If $M \equiv_\alpha N$ then $\mathbb{W}^\sigma_{\lambda xM} = \mathbb{W}^\sigma_{\lambda xN}$.*

The next one is a remarkable result of the present theory. It says that applying a substitution to $\alpha$ convertible terms gives *the same* term.

**Theorem 3.** *If $M \equiv_\alpha N$ then $M\sigma = N\sigma$.*

*Proof.* By induction on $\equiv_\alpha$. There are two interesting cases in our induction. The first is the case of a change of bound variable. Take then $(\lambda xM)\sigma$ and $(\lambda yM\iota[x \mapsto y])\sigma$, with $y \notin FVM$. We have to show them equal. Now notice first that, by virtue of Lemma 2(2), we have $\mathbb{W}^\sigma_{\lambda xM} = \mathbb{W}^\sigma_{\lambda yM\iota[x \mapsto y]}$. Therefore when applying $\sigma$ to both abstractions we will rename the bound variable to the same variable, say $z$. So what we have to show is $\lambda zM\sigma[x \mapsto z] = \lambda z(M\iota[x \mapsto y])\sigma[y \mapsto z]$. Now, applying Corollary 1(2) to the latter we obtain $\lambda zM(\sigma[y \mapsto z])[x \mapsto z]$. But, since $y \notin FVM$, $(\sigma[y \mapsto z])[x \mapsto z] =_M \sigma[x \mapsto z]$, which conducts to the result.

The other case worth looking at is the one corresponding to compatibility of $\equiv_\alpha$ w.r.t. $\lambda$ abstraction. We therefore take $(\lambda xM)\sigma$ and $(\lambda xM')\sigma$ with $M \equiv_\alpha M'$ and show them equal. Now, because of Corollary 2, we know $\mathbb{W}^\sigma_{\lambda xM} = \mathbb{W}^\sigma_{\lambda xM'}$ and therefore the renaming of the bound variable in the effect of $\sigma$ on both abstractions will choose the same variable, say $z$. Thus we have to show $\lambda zM\sigma[x \mapsto z] = \lambda zM'\sigma[x \mapsto z]$, which follows by induction and compatibility of $\equiv_\alpha$ w.r.t. $\lambda z$. □

The preceding theorem is of course also proven in [5] , though the induction is different, due to the different definition of $\equiv_\alpha$. Some of its very nice consequences are explored therein. A remarkable one is that the identity substitution makes an $\alpha$ normal form of any term:

**Corollary 3.** (1) $M \equiv_\alpha N$ *iff* $M\iota{=}N\iota$.
(2) $\sigma \equiv_{\alpha M} \sigma'$ *iff* $\iota \circ \sigma{=}\iota \circ \sigma'$.

The proof is to be found in [5] as are all the remaining ones in this section.
A fundamental result is the following:

**Theorem 4** (Substitution Lemma for $\equiv_\alpha$.)**.** *If* $M \equiv_\alpha N$ *and* $\sigma \equiv_{\alpha M} \sigma'$ *then* $M\sigma{\equiv_\alpha}N\sigma'$.

Its proof is remarkably simple, requiring only calculational reasoning founded upon Lemma 1, Theorem 1, Lemma 4, Theorem 3 and Corollary 3. The following two results will be used in the next section:

**Corollary 4.** (1) *If* $y \notin \mathbb{W}^\sigma_{\lambda xM}$ *then* $(M\sigma[x \mapsto y])\iota[y \mapsto N]{\equiv_\alpha}M\sigma[x \mapsto N]$.
(2) *If* $y \notin \mathbb{W}^\sigma_{\lambda xM}$ *then* $(\lambda xM)\sigma{\equiv_\alpha}\lambda y M\sigma[x \mapsto y]$.

2.5. **Beta reduction.** We finally introduce the notion of computation of $\lambda$ calculus:

A $\beta$ redex is a term $(\lambda xM)N$. A $\beta$ contraction is:

$$(\lambda xM)N \rhd_\beta \ M\iota[x \mapsto N]$$

The *one-step* $\beta$ *reduction* $\rightarrow_\beta$ is the compatible closure of $\rhd_\beta$. The relation of computation, $\beta$ *reduction*, is $\twoheadrightarrow_\beta = (\rightarrow_\beta \cup \equiv_\alpha)*$

## 3. Substitution Lemma for Parallel Reduction

$\beta$ reduction is confluent. A classical method for proving this is Tait–Martin-Löf's, which rests upon proving the confluence of a relation of parallel reduction which we axiomatize here below as an inductive definition. It is essentially the same set of rules as in e.g. [1] but we add a rule allowing explicit $\alpha$ conversions, since we do not identify terms up to that relation.

$$\text{var: } \frac{}{\mathrm{x}{\rightrightarrows}\mathrm{x}}$$

$$\text{abs: } \frac{M \rightrightarrows M'}{\lambda xM \rightrightarrows \lambda xM'}$$

$$\text{app: } \frac{M \rightrightarrows M' \qquad N \rightrightarrows N'}{MN \rightrightarrows M'N'}$$

$$\beta : \frac{M \rightrightarrows M' \qquad N \rightrightarrows N'}{(\lambda xM)N \rightrightarrows M'\iota[x \mapsto N']}$$

$$\alpha : \frac{M \rightrightarrows N \qquad N \equiv_\alpha N'}{M \rightrightarrows N'}$$

We should compare this to its analogue in [4]. In the latter's Appendix A2, a step of parallel reduction is a minimal complete development followed by a number of $\alpha$ conversions. One could then argue that the closest analogue to such definition would be obtained by introducing two mutually defined relations, say $\rightrightarrows_0$ and $\rightrightarrows$, where $\rightrightarrows_0$ reflects the idea of minimal complete development, i.e. it would be given

by the first four rules above with the conclusions appropriately modified, whereas the rule $\alpha$ would turn a $\rightrightarrows_0$ reduction into a $\rightrightarrows$ one. Such a definition, however, can be proven equivalent to the one above.

**Lemma 7.** *If $M \rightrightarrows N$ then $FVN \subseteq FVM$.*

*Proof.* Immediate induction on $\rightrightarrows$. □

We now extend parallel reduction to substitutions in the most direct way, i.e. $\sigma \rightrightarrows \sigma'$ iff $(\sigma\, x) \rightrightarrows (\sigma'\, x)$ for all variables $x$.

**Theorem 5** (Substitution Lemma for $\rightrightarrows$)**.** *If $M \rightrightarrows M'$ then whenever $\sigma \rightrightarrows \sigma'$ we have $M\sigma \rightrightarrows M\sigma'$.*

*Proof.* By induction on $M \rightrightarrows M'$.

(1) Case var: Immediate by definition of $\sigma \rightrightarrows \sigma'$.
(2) Case abs: Take $M \rightrightarrows M'$. The induction hypothesis is that for all $\sigma_1$, $\sigma_1'$ s.t. $\sigma_1 \rightrightarrows \sigma_1'$, $M\sigma_1 \rightrightarrows M'\sigma_1'$. Take now $\sigma$, $\sigma'$ s.t. $\sigma \rightrightarrows \sigma'$. We have to show $(\lambda x M)\sigma \rightrightarrows (\lambda x M')\sigma'$. Now the left hand side is $\lambda y M\sigma[x \mapsto y]$ for appropriately chosen variable $y$. Whatever this $y$ is, we know $\sigma[x \mapsto y] \rightrightarrows \sigma'[x \mapsto y]$. Then, by virtue of the induction hypothesis, we get $M\sigma[x \mapsto y] \rightrightarrows M'\sigma'[x \mapsto y]$ and, by rule abs of $\rightrightarrows$, $\lambda y M\sigma[x \mapsto y] \rightrightarrows \lambda y M'\sigma'[x \mapsto y]$. Now we show that this right hand side is $\alpha$ convertible with $(\lambda x M')\sigma'$, which gives the desired result by using rule $\alpha$ of $\rightrightarrows$. But this is just Corollary 4(2), which we apply by observing that $y \notin \mathbb{W}^{\sigma'}_{\lambda x M'} = \bigcup_{z \in FVM' - \{x\}} FV(\sigma'\, z)$. The latter in turn holds because $y \notin \mathbb{W}^{\sigma}_{\lambda x M} = \bigcup_{z \in FVM - \{x\}} FV(\sigma\, z)$ and both $FVM' \subseteq FVM$ and $FV(\sigma'\, z) \subseteq FV(\sigma\, z)$ by Lemma 7.
(3) Case app: Immediate using induction hypothesis and rule app of $\rightrightarrows$.
(4) Case $\beta$: Take $M \rightrightarrows M'$ and $N \rightrightarrows N'$. The induction hypothesis is that for all $\sigma_1$, $\sigma_1'$ s.t. $\sigma_1 \rightrightarrows \sigma_1'$, $M\sigma_1 \rightrightarrows M'\sigma_1'$ and for all $\sigma_2$, $\sigma_2'$ s.t. $\sigma_2 \rightrightarrows \sigma_2'$, $N\sigma_2 \rightrightarrows N'\sigma_2'$. Take now $\sigma$, $\sigma'$ s.t. $\sigma \rightrightarrows \sigma'$. We have to show $((\lambda x M)N)\sigma \rightrightarrows (M'\iota[x \mapsto N'])\sigma'$ (1). Now the left hand side is $(\lambda y M\sigma[x \mapsto y])N\sigma$ for appropriately chosen variable $y$ (2). We know $\sigma \rightrightarrows \sigma'$ and therefore $\sigma[x \mapsto y] \rightrightarrows \sigma'[x \mapsto y]$. Hence, by the induction hypothesis, $M\sigma[x \mapsto y] \rightrightarrows M'\sigma'[x \mapsto y]$ and $N\sigma \rightrightarrows N'\sigma'$ and then, by rule $\beta$ of $\rightrightarrows$, that left hand side reduces in parallel to $(M'\sigma'[x \mapsto y])\iota[y \mapsto N'\sigma']$. By Corollary 4(1), this is $\alpha$ convertible to $M'\sigma'[x \mapsto N'\sigma']$ (3), because $y \notin FV(\sigma'\, z)$ for $z \in FVM' - \{x\}$. The latter in turn is a consequence of the restriction on $y$ in (2), namely $y \notin FV(\sigma\, z)$ for $z \in FVM - \{x\}$, and the fact that both $FVM' \subseteq FVM$ and $FV(\sigma'\, z) \subseteq FV(\sigma\, z)$ by Lemma 7. Finally, the right hand side of (1) is actually equal to (3) by Corollary 1(1) and thus we get what we wanted by using rule $\alpha$.
(5) Case $\alpha$: Suppose $M \rightrightarrows N$ and $N \equiv_\alpha N'$. The induction hypothesis is that for all $\sigma$, $\sigma'$ s.t. $\sigma \rightrightarrows \sigma'$, $M\sigma \rightrightarrows N\sigma'$. Now take $\sigma$, $\sigma'$ s.t. $\sigma \rightrightarrows \sigma'$. We must show $M\sigma \rightrightarrows N'\sigma'$. But, since $N \equiv_\alpha N'$, $N'\sigma' = N\sigma'$ by Theorem 3, and by induction hypothesis $M\sigma \rightrightarrows N\sigma'$, thus getting what was required.

□

The definition of parallel reduction between substitutions can be made relative to the free variables of a term, in the same way as we have done in the preceding section

with equality and $\alpha$ conversion. Then the substitution lemma can be formulated accordingly and the method of proof carries over.

After the substitution lemma, confluence of parallel reductions and thereby of $\beta$ reduction are quite direct.

## 4. Substitution Lemma for Type Assignment

4.1. **Types.** Let $\iota$ be a class of ground types. Then the set $\mathbb{A}$ of simple types $\alpha$ are defined as follows:

$$\alpha :- \iota | \alpha \to \beta$$

4.2. **Contexts.** Contexts $\Gamma$ are finite functions $\mathbb{V} \to \mathbb{A}$.

We write $\Gamma - x$ for $\Gamma \restriction (dom\,\Gamma - \{x\})$

We use from now on the following differentiated notations for function overriding:

$\Gamma \cdot x : \alpha$ will be used for context overriding, and

$\sigma[x := M]$ for substitution overriding.

4.3. **Rules of Type Assignment.**

$$\text{var: } \frac{}{\Gamma \vdash x : \Gamma\, x}\ x \in dom\,\Gamma$$

$$\text{app: } \frac{\Gamma \vdash M : \alpha \to \beta \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}$$

$$\text{abs: } \frac{\Gamma \cdot x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x M : \alpha \to \beta}$$

In [3, Ch. 2, Sect. 2A] a different formulation is given, that can however be proven equivalent to the present one.

The following structural properties of the system hold.

**Lemma 8** (Weakening)**.** *If $\Gamma \vdash M : \alpha$ and $\Gamma \subseteq \Delta$ then $\Delta \vdash M : \alpha$.*

*Proof.* Easy induction on $\Gamma \vdash M : \alpha$. □

**Lemma 9** (Strengthening)**.** *If $\Gamma \vdash M : \alpha$ and $x \notin FV\,M$ then $\Gamma - \{x\} \vdash M : \alpha$.*

*Proof.* Induction on $\Gamma \vdash M : \alpha$. □

**Corollary 5.** *If $\Gamma \vdash M : \alpha$ and $x \notin FV\,M$ then $\Gamma \cdot x : \beta \vdash M : \alpha$.*

*Proof.* If $\Gamma \vdash M : \alpha$ and $x \notin FV\,M$ then $\Gamma - \{x\} \vdash M : \alpha$ by Strengthening. Now $\Gamma - \{x\} \subseteq \Gamma \cdot x : \beta$ and then the conclusion follows by Weakening. □

Typing is extended to substitutions, in the following way:

$\sigma : \Gamma \to \Delta$ iff for each $x \in dom\,\Gamma$, $\Delta \vdash \sigma\, x : \Gamma\, x$.

An important consequence of Corollary 5 is the following:

**Corollary 6.** *If $\sigma : \Gamma \to \Delta$ and $y \notin FV(\sigma\, z)$ for $z \in dom\,\Gamma - \{x\}$ then $\sigma[x := y] : \Gamma \cdot x : \alpha \to \Delta \cdot y : \alpha$.*

*Proof.* Take variable $z \in dom\,(\Gamma \cdot x : \alpha)$. If $z = x$ then $(\sigma[x := y])\, z = y$ and $\Delta \cdot y : \alpha \vdash y : \alpha$, where we have $\alpha = (\Gamma \cdot x : \alpha)\, z$. If $z \neq x$ then $(\sigma[x := y])\, z = \sigma\, z$. Now, since $\sigma : \Gamma \to \Delta$, we know $\Delta \vdash \sigma\, z : \Gamma\, z$ and, because $y \notin FV(\sigma\, z)$, we can apply Corollary 5 and get $\Delta \cdot y : \alpha \vdash \sigma\, z : \Gamma\, z$. Besides, $\Gamma\, z = (\Gamma \cdot x : \alpha)\, z$. Thus in any case $\Delta \cdot y : \alpha \vdash (\sigma[x := y])\, z : (\Gamma \cdot x : \alpha)\, z$, as required. □

Thus we arrive to our final result:

**Theorem 6** (Substitution Lemma for Type Assignment)**.** *If* $\Gamma \vdash M : \alpha$ *and* $\sigma : \Gamma \to \Delta$ *then* $\Delta \vdash M\sigma : \alpha$.

*Proof.* Easy induction on $\Gamma \vdash M : \alpha$. We spell it out in detail just for completeness: For the case (var:), we get $x \in dom\,\Gamma$ and therefore $\Delta \vdash \sigma\, x : \Gamma\, x$. But $\sigma\, x{=}x\sigma$ and we have what is required. The case (app:) is immediate by induction. Finally, for the case (abs:), observe first that $(\lambda xM)\sigma{=}\ \lambda yM\sigma[x \mapsto y]$ where $y \notin FV(\sigma\, z)$ for $z \in FVM - \{x\}$. From the latter and $\sigma : \Gamma \to \Delta$ it follows by Corollary 6 that $\sigma[x := y] : \Gamma \cdot x : \alpha \to \Delta \cdot y : \alpha$. Hence by induction $\Delta \cdot y : \alpha \vdash M\sigma[x := y] : \beta$ and thereby, using rule (abs:) of type assignment, $\Delta \vdash \lambda yM\sigma[x := y] : \alpha \to \beta$, as required. □

Once this is established it is just straightforward to prove the important results of subject reduction (under both $\alpha$ and $\beta$ rules).

## 5. Conclusion

The first conclusion we wish to make is that we have identified strong reasons to consider Stoughton's approach, i.e. that of considering long, simultaneous substitutions and uniform renaming of the bound variable when they effect on abstractions, as the right one for formulating the lambda calculus in its original syntax, i.e. with named variables and no differentiation between symbols to appear free or bound.

To this we have arrived after developing the theory of substitution and $\alpha$ conversion in detail, without identifying $\alpha$ convertible terms or using Barendregt's convention. This theory is already exposed in Stoughton's work [5], but we wanted to give an alternative formulation which we think is (if slightly) more natural, besides it being more detailed. The fundamental difference between both presentations resides in the definition of alfa conversion. He makes it with basis on a rule of lambda abstraction with two subcases, whereas we generate it straightforwardly from the simple operation of change of bound variable, the same way as in [3, 4]. As an additional difference, we develop as much of the theory of substitutions that is possible before introducing $\alpha$ conversion, whereas [5] introduces the latter from the beginning. As a result, we prove Lemma 1 without mention to $\alpha$ conversion; in [5] the proof seems to require a previous result on $\alpha$ convertible terms, although not enough detail is given to tell conclusively.

Now, turning back to our main conclusion at the beginning, the theory of substitution and $\alpha$ conversion covers everything that is proven in [4, 2], but runs much more smoothly and with better properties. This is due to the much simpler definition of the effect of substitution on terms. First, the long substitutions allow for structural recursion since the change of bound variable needed in cases of capture can be recorded in the substitution to be applied. Secondly, the treatment of substitution on lambda abstractions, uniformly effecting a change of the bound variable, simplifies the reasoning about such cases, avoiding consideration of multiple subcases. The result is that, e.g. the identity substitution on a term produces an $\alpha$ equivalent term and not the term itself (Lemma 4). But working uniformly with $\alpha$ conversion turns out to be simpler than trying to do with identity in as many cases as possible, given of course the fact that the latter is not *always* possible. The fundamental insight in this respect is that the condition to avoid capture can be given in a simple way (i.e. the definition of the set $\mathbb{W}^{\sigma}_{\lambda xM}$ of the variables to be excluded for the choice of new bound variable) which depends on the free variables

of the abstraction and the free variables of their images in the substitution to be applied.

As consequences of these simplifications, we have first that the proofs of the theory of substitution and $\alpha$ conversion are by structural induction on terms instead of complete induction on the length of terms. The traditional formulation needs the latter already from the beginning for justifying the recursion employed in the definition of substitution. Although the fact is never mentioned in the classic textbooks [2, 3, 4], such justification must be developed in simultaneity with a proof that substitution of a variable for a variable does not change the size of the term. As a second, interesting consequence related to the way substitution is effected, we have Theorem 4 of identity of effect of a substitution on $\alpha$ convertible terms and its corollary of $\alpha$ normalization of terms via identity substitution.

With respect to the substitution lemma for parallel reduction, we have achieved a direct inductive proof without provisos justified by lemmas postponing $\alpha$ conversions, i.e. without having to previously manipulate the structure of the presupposed computations, as is done in [4]. The results allowing this are quite natural: Corollary 4(1), which is a converse of Corollary 1(2), which in turn is a direct consequence of distribution of composition over overriding. Corollary 1(2) is also proven in [4], but much more awkwardly.

Finally, manipulations of the structure of reductions are needed in [3] also for proving the substitution lemma for type assignment, in order to avoid the case of renaming in substitution. Thus the substitution lemma takes a quite complicated structure, with a particular case first, then the proof of closure under $\alpha$ conversion and then the general case, manipulating previously the structure of the deduction. In our approach, Corollary 6 establishes a simple and powerful correspondence between substitution and context overriding precisely when introducing allowable bound variable in renaming. This makes the whole proof extremely simple.

As a final conclusion, we have established the feasibility of the development of the basic theory of lambda calculus using its original syntax, working directly with terms and not $\alpha$ equivalence classes thereof. In particular, this sets an alternative for full formalisations of the lambda calculus and its theory in e.g. type theory as implemented in the available proof assistants. This remains as work to be done.

## References

[1] H.P. Barendregt. *The Lambda Calculus: its Syntax and Semantics.* (North-Holland, Amsterdam, 1984.)
[2] H.B. Curry, R. Feys. *Combinatory Logic, Vol. I.* (North-Holland, Amsterdam, 1958.)
[3] J.R. Hindley. *Basic Simple Type Theory.* (Cambridge University Press, 1997.)
[4] J.R. Hindley, J. Seldin. *Lambda-Calculus and Combinators, an Introduction.* (Cambridge University Press, 2008.)
[5] A. Stoughton. *Substitution Revisited.* Theoretical Computer Science 59 (1988) 317-325.

Universidad ORT Uruguay
*E-mail address*: tasistro@ort.edu.uy