Vol. 42 No. 1 IPSJ Journal Jan. 2001

Regular Paper

A Type-Free Context Calculus

Azza A. Taha,† Masahiko Sato† and Yukiyoshi Kameyama†

This paper develops a type free context calculus λxc. The calculus λxc includes contexts as first-class values and hole-filling as an explicit operation. In λxc, holes are represented by ordinary variables and hole-filling is represented by the usual application together with a new abstraction mechanism which represents the variables intended to be bound after filling in the hole. We show that this calculus has desirable properties such as confluence, conservativity over $\lambda\beta$ calculus and has the preservation of strong normalization (PSN) property.

1. Introduction

In the lambda calculus, a context is a term with some holes in it, e.g., $\lambda x.[.]$, where [.] indicates a hole. The basic operation for a context is to fill its holes with a term. For example, filling the hole above with the term $x + y$ results in $\lambda x.x + y$, in which x is captured by λ. So, unlike capture-avoiding substitution in the lambda calculus, hole-filling may introduce new and intended bound variables.

In this paper, we introduce the calculus λxc as an extension of the calculus λx defined by Bloo and Rose[2]. In addition to the terms of λx, we added two new terms to be able to compute with contexts. In our calculus, holes are represented by ordinary variables and hole-filling is represented by the usual application together with a new abstraction mechanism which represents the variables intended to be bound after filling in the hole. It is clear that holes are first-class objects in this calculus, so we can pass it to and return it from a function. This idea of representing contexts is due to Sato, et al.[8]. They extend the explicit environment calculus $\lambda\varepsilon$[9], which contains environments as first-class objects, to include contexts also as first-class objects. Our system is different from Sato, et al.'s system[8] in the following four points:

(1) Our system is untyped so it is allowed to consider expressions like FF, i.e., F applied to itself, and so there will be a lot of freedom in combining terms.
(2) The system defined by Sato, et al.[8] is based on explicit environments. However, to treat contexts it is sufficient to use explicit substitutions which is simpler than explicit environments. So, we adopted explicit substitutions instead of environments.
(3) Another point which makes our system simpler than that of Sato, et al.'s system is that we restrict the variables bound by λ as well as the variables bound by the set $\{\overline{x}\}$ to pure variables—variables without #'s—so we do not have to introduce more complex operations.
(4) The reduction in our calculus respects the α-equivalence, while this does not hold in Sato, et al.'s system due to the presence of the environment type.

In general, when computing with contexts two problems arise.

First, α-conversion and hole-filling do not commute. To see this problem, consider the term $\lambda x.[.]$ which is α-equivalent to $\lambda y.[.]$. After filling these two terms with a variable x, we get $\lambda x.x$ and $\lambda y.x$, respectively, which are not α-equivalent.

We solve this problem by writing the above terms as $\lambda x.(X \bullet [\langle x := x\rangle])$ and $\lambda y.(X \bullet [\langle x := y\rangle])$ where X represents a hole. If we fill this hole X with the variable x, we substitute the new abstraction $\{x\}.x$ for it, which dictates that the variable x is intended to be bound, and we get:

$$\lambda x.(\{x\}.x) \bullet [\langle x := x\rangle] \equiv \lambda x.x, \text{and}$$
$$\lambda y.(\{x\}.x) \bullet [\langle x := y\rangle] \equiv \lambda y.y$$

which are α-equivalent.

Second, β-reduction and hole-filling also do not commute. For example, in the term $(\lambda z.[.])y$, if the hole in it is filled first with a variable z and then the β-reduction is applied, we get y, while if the β-reduction is applied first and then the variable z is filled in the hole we get z, and the results are not the same.

In our calculus, the above term is written as:

† Graduate School of Informatics, Kyoto University

$(\lambda X.(\lambda z.X \bullet [\langle z := z\rangle])y)(\{z\}.z)$

If the hole in this term is filled first and then the β-reduction is applied, or the β-reduction is applied and then the term is filled in the hole, we get the same result y.

Also, note that the usual α-conversion cannot be used here to rename the variables bound by the $\{x\}$ binder for the following reason:

Consider the following program taken from Ref. 8):

$((\{x\}.x + y)\langle y := x\rangle) \bullet [\langle x := 2\rangle]$

If α-conversion is applied here and then we substitute x for y we get:

$$\stackrel{\alpha}{\to} ((\{z\}.z + y)\langle y := x\rangle) \bullet [\langle x := 2\rangle]$$
$$\to (\{z\}.z + x) \bullet [\langle x := 2\rangle]$$

which cannot be further reduced since the variable x in $\langle x := 2\rangle$ does not match the binder $\{z\}$.

To remedy this problem, we adapt the method defined by Sato, et al.[8] to rename free variables (if necessary). We can compute the above term as follows:

$(\{x\}.x + \#x) \bullet [\langle x := 2\rangle] \to 2 + x$

Recently, there have been several attempts to formalize and compute with contexts. However, this is the first untyped context calculus which has contexts as first-class values, hole-filling as an explicit operation and which is at the same time pure in the sense of Ref. 9), i.e., (i) it is a conservative extension of the untyped $\lambda\beta$-calculus, (ii) confluent and (iii) has the preservation of strong normalization (PSN) property.

There are also a lot of calculi which made notable contribution in this field:

Sands[7] outlines the use of higher order syntax to represent and compute with contexts. In his approach, holes are represented by an application of some function variable to a vector of variables. Each function variable has a given fixed arity, which dictates exactly how many arguments it expects. His representation of a context can be α-converted in the usual manner. Hole-filling is represented by substituting a meta-abstraction for this variable. In his approach, substitution commutes with hole-filling and there is no problem with the α-equivalence. However, since hole-filling is a meta level operation in his system, it is not possible to compute hole-filling within his system. This approach is reministic of Klop's Combinatory Reduction Systems[5].

Mason[6] provides a representation of contexts within the framework of the λ-calculus, and shows how one can compute with such contexts using this representation. However, as in Sands' approach, although holes are first-class objects in his system, hole-filling is a meta level operation.

Hashimoto-Ohori[4] develop a typed calculus which has contexts as first-class objects and hole-filling as an explicit operation. However, in their system β-reduction is restricted to those redices that do not contain holes.

Dami[3] defines a calculus λ_N, in which the representation of both contexts and hole-filling is possible. However, he did not define any formal system for it, and this representation is done by a translation from $\lambda\beta$ to λ_N.

The rest of the paper is organized as follows. In Section 2, we introduce the system λxc. In Section 3, we prove that λxc has a number of desirable properties such as confluence, conservativity over $\lambda\beta$ calculus and PSN, and also we define the α-equivalence and the meta-level substitution. Finally, the conclusion is given in Section 4.

2. The System

2.1 Terms

Notation $\overline{\langle x := N\rangle}$ will be used to abbreviate $\langle x_1 := N_1\rangle, \ldots, \langle x_n := N_n\rangle$, $\overline{x}$ to abbreviate $x_1, \ldots, x_n$, and $\overline{x_{n,i}}$ to abbreviate $x_n, \ldots, x_i$ for $n \geq i \geq 0$.

Assume that P is the set of *pure variables* $x, y, z, \ldots$, and V is the set of *variables* given by $V ::= P|\#V$. We will use $u, v, w, \ldots$ as meta-variables for variables. As a shorthand we will write $\#^0 x = x$ and $\#^{n+1} x = \#(\#^n x)$. Assume that *pure* is a function which takes a variable and returns its pure version, e.g., pure$(x) = x$ and pure$(\#y) = y$. *Terms* are given by the following grammar.

$$M, N ::= V \mid \lambda x.M \mid MN \mid M\langle v := N\rangle$$
$$\mid \{\overline{x}\}.M \mid M \bullet [\overline{\langle x := N\rangle}]$$

Note that, the order of the variables in the set $\{\overline{x}\}$ is irrelevant, in contrast to the order of $\langle x_i := N_i\rangle$ in the term $M \bullet [\overline{\langle x := N\rangle}]$ is important as we have to reduce it in the order it is given to achieve confluence. This is another point which differentiates this system from the system defined in Ref. 8), which has environment instead of this sequence of substitutions, and this environment is evaluated simultaneously.

Also note that we have the following notions of boundness of variables. The variables x, v and $x_1, \cdots, x_n$ are considered bound in M in the terms $\lambda x.M$, $M\langle v := N\rangle$ and $\{x_1, \cdots, x_n\}.M$, respectively. Besides, in the term $M \bullet [\langle x_1 := N_1\rangle, \ldots, \langle x_n := N_n\rangle]$, if x_j is in the set of free variables of N_i for $j > i$, then they are considered bound. For example, in the term $M \bullet [\langle x_1 := x_2 y\rangle, \langle x_2 := N_2\rangle]$, the variable x_2 in the first substitution is considered bound while y is free.

2.2 The Push and Pull Operators

Before giving the formal definition, some explanation is necessary to clarify the motivation behind using the push and pull operators.

Suppose we have the term $(\lambda x.\lambda y.y + x)y$. It is known that before reducing this term, the α-conversion is necessary to avoid the unwanted capture of y by the inner λ. Thus, if y is renamed to e.g., z, we get $\lambda z.z + y$ as a correct result of reducing the above term. For the reason explained in the introduction, the α-conversion is no longer appropriate for our calculus. Instead, the push operator is used to rename the free variables that would otherwise be captured. For the above term, the free variable y is renamed to $\#y$ and we get $\lambda y.y + \#y$ as a result of reducing the above term. Note that in this term the variable $\#y$ is different from y. Suppose now that 3 is substituted for y in this term, then $\#y$ will no longer be in the scope of λ and we have to pull one $\#y$ out of y and get $3 + y$. It is clear that when there is no collision with the other existing variables, there is no need to rename the free variable. For example, $(\lambda y.\lambda z.y)(z + x)$ is reduced to $\lambda z.\#z + x$.

The relation $\leq$ on variables is defined as: $v \leq w$ iff $w = \#^n v$ for some $n \in N$ or $\mathrm{pure}(v) \neq \mathrm{pure}(w)$. It is clear that for any two variables v, w, we have $v \leq w$ or $w \leq v$.

The following definitions, "push M through v" and "pull M from v" written as $M \uparrow v$ and $M \downarrow v$ respectively were invented by Sato, et al.[8]. Here it is modified to fit the terms in λxc.

$w \uparrow v$ is defined as:

(1) $w \uparrow v = \#w$ if $v \leq w$ and $\mathtt{pure}(w) = \mathtt{pure}(v)$, and

(2) $w \uparrow v = w$, otherwise.

Let E be a finite set of variables, then $w \uparrow E$ is defined inductively on the number of elements in E as follows:

(1) If $E = \phi$, then $w \uparrow E = w$.

(2) Otherwise, $\exists F \exists v$ s.t. $E = F \cup \{v\}$, where $\forall v' \in F,\ v' \leq v$ and $w \uparrow E = (w \uparrow F) \uparrow v$.

Note that, in (2) above when the set E contains different pure variables, the choice of v in $F \cup \{v\}$ is not uniquely determinated, but still the final result is unique. For example, the order of pushing $\#^2 x$ through the elements in the set $\{\#x, y\}$ is not important and we always get $\#^3 x$ as a correct result.

The operation $\downarrow$ is defined as the inverse partial function of $\uparrow$. The term $w \downarrow E$ is defined only when $w \notin E$.

Example 1

(1) If $\mathtt{pure}(x) \neq \mathtt{pure}(y)$, then $x \uparrow \{x, \#x, y\} = \#^2 x$, and $x \uparrow \{\#x, y\} = x$.

(2) $\#^2 x \downarrow \{x, \#x\} = x$ and $\#^3 x \downarrow \{x, \#^3 x\}$ is undefined.

Let $F = \{v_1, \ldots, v_n\}$. We put $F \uparrow E = \{v_1 \uparrow E, \ldots, v_n \uparrow E\}$.

Let M be a term and E, F be finite sets of variables, then the operation $M \uparrow_F E$ is defined inductively as follows. (When F is empty and M is a variable then this definition coincides with the previous one.) Suppose that we wished to push $M\langle v := N\rangle$ through E then we have to keep free occurrences of v in M as they are, since they are bound by v. Namely, what we have to do is to push $M\langle v := N\rangle$ through E *keeping* free occurrences of M. So, in general we will need a set F of variables which must be kept in the process of pushing $M\langle v := N\rangle$ through E. For example $(v\langle v := N\rangle) \uparrow v = v\langle v := N \uparrow v\rangle$ and $(v\langle \#v := N\rangle) \uparrow v = \#^2 v\langle \#v := N \uparrow v\rangle$.

(1) $u \uparrow_F E = \begin{cases} u & \text{if } u \in F, \\ ((u \downarrow F) \uparrow E) \uparrow F & \text{o.w.} \end{cases}$

(2) $(\lambda z.M) \uparrow_F E = \lambda z.(M \uparrow_{(F \uparrow z)} (E \uparrow z))$.

(3) $(MN) \uparrow_F E = (M \uparrow_F E)(N \uparrow_F E)$.

(4) $(M\langle v' := N\rangle) \uparrow_F E = (M \uparrow_{((F \uparrow v') \cup v')} E)\langle v' := N \uparrow_F E\rangle$.

(5) $(\{\overline{x}\}.M) \uparrow_F E = \{\overline{x}\}.(M \uparrow_{(F \uparrow \{\overline{x}\})} (E \uparrow \{\overline{x}\}))$.

(6) $(M \bullet [\overline{\langle x := N\rangle}]) \uparrow_F E = (M \uparrow_F E) \bullet [\langle x_1 := (N_1 \uparrow_{(F \uparrow \overline{x_{n,2}})} (E \uparrow \overline{x_{n,2}}))\rangle \ldots \langle x_n := (N_n \uparrow_F E)\rangle]$.

When F is empty, we write $M \uparrow E$ for $M \uparrow_F E$.

Note that, $M \uparrow x_n \uparrow x_{n-1} \ldots \uparrow x_i$ stands for $(((M \uparrow x_n) \uparrow x_{n-1}) \ldots \uparrow x_i)$.

The operation $\downarrow$ is defined as the inverse partial function of $\uparrow$. $M \downarrow E$ is defined only when $\mathtt{FV}(M) \cap E = \phi$.

The set of free variables in a term M, written as $\mathtt{FV}(M)$ is defined as:

(1) $\mathtt{FV}(v) = \{v\}$.

(2) $\mathtt{FV}(\lambda x.M) = \mathtt{FV}(M) - \{x\}$.

(3) $\mathtt{FV}(MN) = \mathtt{FV}(M) \cup \mathtt{FV}(N)$.
(4) $\mathtt{FV}(M\langle v := N\rangle) = (\mathtt{FV}(M) - \{v\}) \cup \mathtt{FV}(N)$.
(5) $\mathtt{FV}(\{\overline{x}\}.M) = \mathtt{FV}(M) - \{\overline{x}\}$.
(6) $\mathtt{FV}(M \bullet [\langle x_1 := N_1\rangle, \ldots, \langle x_n := N_n\rangle]) = \mathtt{FV}(M) \cup (\mathtt{FV}(N_1) - \{\overline{x_{n,2}}\}) \cup (\mathtt{FV}(N_2) - \{\overline{x_{n,3}}\}) \cup \ldots \cup \mathtt{FV}(N_n)$.

where the above minus operator is defined below.

Let w be a variable and E be $\{v_1, \ldots, v_n\}$, V be $\{x_1, \ldots, x_m\}$, respectively. We also put E_1 be $E - \{v_n\}$, that is, $\{v_1, \ldots, v_{n-1}\}$. Then we define a set of variables $E - V$ as follows:

$$(1)\ E - \{w\} = \begin{cases} \phi & \text{if } E = \phi, \\ E_1 - \{w\} & \text{if } v_n = w, \\ (E_1 - \{w\}) \cup \{v_n \downarrow w\} & \text{o.w.} \end{cases}$$

(2) $E - V = ((E - \{x_1\}) \cdots - \{x_m\})$.

Note that, in (2) above any order of subtracting the x_i's will give the same result as it is a sequence of distinct pure variables.

Example 2 $\mathtt{FV}((\{x, y\}.(x+y)) \bullet [\langle x := \#^3 y\rangle \langle y := N\rangle]) = \{\#^2 y\} \cup \mathtt{FV}(N)$.

2.3 Reductions

The *reduction rules* of λxc are the union of the following 3 relations $\to_{\mathrm{b}}$, $\to_{\mathrm{C}}$ and $\to_{\mathrm{x}}$.

The relation $\to_{\mathrm{b}}$ is defined by the following rule:

$$(b)\quad (\lambda x.M)N \to_{\mathrm{b}} M\langle x := N\rangle.$$

The relation $\to_{\mathrm{C}}$ is defined by the following rule:

$$(c)\quad (\{\overline{x}\}.M) \bullet [\overline{\langle x := N\rangle}] \to_{\mathrm{C}} M\overline{\langle x := N\rangle}$$

The relation $\to_{\mathrm{x}}$ is defined by the following 6 rules:

$(\mathtt{x}var)$ $v\langle v := N\rangle \to_{\mathrm{x}} N$.

(gc) $M\langle v := N\rangle \to_{\mathrm{x}} M \downarrow v$ if $v \notin FV(M)$.

$(\mathtt{x}abs)$ $(\lambda x.M)\langle v := N\rangle \to_{\mathrm{x}} \lambda x.M\langle v \uparrow x := N \uparrow x\rangle$.

$(\mathtt{x}app)$ $(M_1 M_2)\langle v := N\rangle \to_{\mathrm{x}} (M_1\langle v := N\rangle)(M_2\langle v := N\rangle)$.

(abs) $(\{\overline{x}\}.M)\langle v := N\rangle \to_{\mathrm{x}} \{\overline{x}\}.(M\langle v \uparrow \{\overline{x}\} := N \uparrow \{\overline{x}\}\rangle)$.

(app) $(M \bullet [\overline{\langle x := N\rangle}])\langle v := P\rangle$
$\to_{\mathrm{x}} M\langle v := P\rangle \bullet$
$[\langle x_1 := N_1\langle v \uparrow \overline{x_{n,2}} := P \uparrow \overline{x_{n,2}}\rangle\rangle$
$\cdots$
$\langle x_n := N_n\langle v := P\rangle\rangle]$.

Note that, in the (c) rule, although the order of the variables in the set $\{\overline{x}\}$ is irrelevant, they must match the variables x_i's which appear in the sequence of the substitution.

We write $M \to_{\mathrm{b}} N$ if N is obtained from M by replacing a subterm M_1 in M by N_1 such that $M_1 \to_{\mathrm{b}} N_1$. Similarly $\to_{\mathrm{C}}$, $\to_{\mathrm{x}}$ and $\to_{\mathrm{bxc}}$ are defined. The reflexive and transitive closure of the $\to$ reduction is denoted by $\overset{*}{\to}$. $\to_{\mathrm{bc}}$ is the union of the two reduction relations $\to_{\mathrm{b}}$ and $\to_{\mathrm{C}}$.

Example 3 Consider the following example taken from Hashimoto-Ohori's paper [4]:

$$(\lambda y.(\delta X.(\lambda x.X)3) \odot (x+y))x$$

In our system, this term can be written as:

$$(\lambda y.(\lambda X.(\lambda x.X \bullet [\langle x := x\rangle])3)(\{x\}.(x+y)))x$$

Let M be $\{x\}.(x+y)$. Then the above term can be computed as follows:

$$\begin{array}{ll}
\to_{\mathrm{b}} & (\lambda y.(\lambda X.(X \bullet [\langle x := x\rangle])\langle x := 3\rangle)M)x \\
\overset{*}{\to}_{\mathrm{x}} & (\lambda y.(\lambda X.(X \bullet [\langle x := 3\rangle]))M)x \\
\to_{\mathrm{b}} & (\lambda X.(X \bullet [\langle x := 3\rangle])M)\langle y := x\rangle \\
\overset{*}{\to}_{\mathrm{x}} & (\lambda X.(X \bullet [\langle x := 3\rangle])(\{x\}.(x + \#x))) \\
\to_{\mathrm{b}} & (X \bullet [\langle x := 3\rangle])\langle X := \{x\}.(x + \#x)\rangle \\
\overset{*}{\to}_{\mathrm{x}} & \{x\}.(x + \#x) \bullet [\langle x := 3\rangle] \\
\to_{\mathrm{C}} & (x + \#x)\langle x := 3\rangle \\
\overset{*}{\to}_{\mathrm{x}} & 3 + x.
\end{array}$$

Example 4 The following example is taken from Ref. 8). Let N_1 be $\{x, y\}.(x+y)$, N_2 be $X \bullet [\langle x := x\rangle, \langle y := y\rangle]$, and N_3 be $N_1 \bullet [\langle x := x\rangle, \langle y := y\rangle]$.

$$\begin{array}{ll}
 & (\lambda X.\lambda x.(\lambda y.(x + N_2))3)N_1 \\
\to_{\mathrm{b}} & (\lambda x.(\lambda y.(x + N_2))3))\langle X := N_1\rangle \\
\overset{*}{\to}_{\mathrm{x}} & \lambda x.(\lambda y.(x + N_3))3 \\
\to_{\mathrm{b}} & \lambda x.(x + N_3)\langle y := 3\rangle \\
\overset{*}{\to}_{\mathrm{x}} & \lambda x.(x + N_1 \bullet [\langle x := x\rangle, \langle y := 3\rangle]) \\
\to_{\mathrm{C}} & \lambda x.x + ((x+y)\langle x := x\rangle\langle y := 3\rangle) \\
\overset{*}{\to}_{\mathrm{x}} & \lambda x.x + (x+3).
\end{array}$$

Thus, we have much freedom of reductions.

3. Properties of λxc

In this section, we show that λxc has a number of desirable properties. First, we prove the confluence.

3.1 Confluence

Lemma 1 The relation $\to_{\mathrm{x}}$ on λxc-terms is noetherian and confluent.

Proof In order to show that $\to_{\mathrm{x}}$ is noetherian, the length $|M|$ is introduced, which is a positive integer defined for each term as follows:

(1) $|v| := 1$.
(2) $|\lambda x.M| := |M| + 1$.
(3) $|MN| := |M| + |N| + 1$.
(4) $|M\langle v := N\rangle| := (|N| + 1)|M|$.
(5) $|\{\overline{x}\}.M| := |M| + 1$.
(6) $|M \bullet [\langle x_1 := N_1\rangle \ldots \langle x_n := N_n\rangle]| := |M| + |N_1| + \ldots + |N_n| + 1$.

Then, it can easily be verified that if $M \to_{\mathrm{x}}$

N then $|M| > |N|$. By checking the overlapping cases, it can easily be verified that $\to_{\mathrm{x}}$ on λxc is weakly Church-Rosser. Combining these two, we have confluence by Newman's lemma. $\square$

A term M is x-*normal* if $M \to_{\mathrm{x}} N$ holds for no N. By the above lemma, it is clear that for any term M, there uniquely exists an x-normal term N s.t. $M \stackrel{*}{\to}_{\mathrm{x}} N$. We will write $\mathtt{x}(M)$ for this N. The x-normal terms are characterized by the following grammar where c ranges over x-normal terms:

$$c ::= v \mid \lambda x.c \mid cc \mid \{\overline{x}\}.c \mid c \bullet [\overline{\langle x := c\rangle}]$$

The *parallel reduction relation* $\Rightarrow$ on x-normal terms is defined as:

(1) $v \Rightarrow v$.
(2) If $M \Rightarrow N$, then $\lambda x.M \Rightarrow \lambda x.N$.
(3) If $M_1 \Rightarrow M_2$ and $N_1 \Rightarrow N_2$, then $(\lambda x.M_1)N_1 \Rightarrow \mathtt{x}(M_2\langle x := N_2\rangle)$.
(4) If $M_1 \Rightarrow M_2$ and $N_1 \Rightarrow N_2$, then $M_1N_1 \Rightarrow M_2N_2$.
(5) $M \Rightarrow N$, then $\{\overline{x}\}.M \Rightarrow \{\overline{x}\}.N$.
(6) If $M \Rightarrow N$ and $P_i \Rightarrow Q_i, 1 \le i \le n$ then $(\{\overline{x}\}.M) \bullet [\langle x_1 := P_1\rangle \ldots \langle x_n := P_n\rangle] \Rightarrow \mathtt{x}(N\langle x_1 := Q_1\rangle \ldots \langle x_n := Q_n\rangle)$.
(7) If $M \Rightarrow N$ and $P_i \Rightarrow Q_i, 1 \le i \le n$ then $M \bullet [\langle x_1 := P_1\rangle, \ldots, \langle x_n := P_n\rangle] \Rightarrow N \bullet [\langle x_1 := Q_1\rangle \ldots \langle x_n := Q_n\rangle]$.

Next, with each x-normal term M, we associate an x-normal term M^* as follows:

(1) $v^* := v$.
(2) $(\lambda x.M)^* := \lambda x.M^*$.
(3) $((\lambda x.M)N)^* := \mathtt{x}(M^*\langle x := N^*\rangle)$.
(4) $(MN)^* := M^*N^*$, if M is not a λ abstraction.
(5) $(\{\overline{x}\}.M)^* := \{\overline{x}\}.M^*$.
(6) $((\{\overline{x}\}.M) \bullet [\langle x_1 := P_1\rangle \ldots \langle x_n := P_n\rangle])^* := \mathtt{x}(M^*\langle x_1 := P_1^*\rangle \ldots \langle x_n := P_n^*\rangle)$.
(7) $(M_1 \bullet [\langle x_1 := P_1\rangle \ldots \langle x_n := P_n\rangle])^* := M_1^* \bullet [\langle x_1 := P_1^*\rangle \ldots \langle x_n := P_n^*\rangle]$, if M_1 is not in the form $\{\overline{x}\}.M$.

It can easily be verified that $M \Rightarrow M$ and $M \Rightarrow M^*$ hold for any x-normal term M.

The following lemma is important in proving Theorem 1.

Lemma 2 If $M \Rightarrow N$, then $M \to_{\mathrm{bxc}}^* N$.

Proof By induction on the construction of M.

We need the following lemma in proving Lemma 6.

Lemma 3 If $\mathtt{x}(M) \Rightarrow \mathtt{x}(M')$ then we have $\mathtt{x}(M \uparrow v) \Rightarrow \mathtt{x}(M' \uparrow v)$.

Proof By induction on the construction of $\mathtt{x}(M)$.

The following lemma is important to prove Corollary 1, which is important to prove Lemma 6.

Lemma 4 (Substitution lemma)
If M is an x-normal term, then we have
$\mathtt{x}(M\langle v := N\rangle\langle w := O\rangle) \equiv$
$\mathtt{x}(M\langle w \uparrow v := O \uparrow (v \downarrow (w \uparrow v))\rangle\langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle)$

Proof We shall prove this lemma by induction on the construction of M. Since M is x-normal, we have the following cases.

(1) M is a variable.
(a) $M \equiv v$.

$$\begin{aligned} LHS &\equiv \mathtt{x}(v\langle v := N\rangle\langle w := O\rangle) \\ &\equiv \mathtt{x}(N\langle w := O\rangle) \quad \text{(by (x}var\text{))}. \\ RHS &\equiv \mathtt{x}(v\langle w \uparrow v := O \uparrow (v \downarrow (w \uparrow v))\rangle \\ &\qquad \langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle) \\ &\equiv \mathtt{x}((v \downarrow (w \uparrow v))\langle v \downarrow (w \uparrow v) := \\ &\qquad N\langle w := O\rangle\rangle) \quad \text{(by } (gc)) \\ &\equiv \mathtt{x}(N\langle w := O\rangle) \quad \text{(by (x}var\text{))}. \end{aligned}$$

(b) $M \not\equiv v$, $M \downarrow v \equiv w$ i.e., $w \uparrow v \equiv M$.

$$\begin{aligned} LHS &\equiv \mathtt{x}(M\langle v := N\rangle\langle w := O\rangle) \\ &\equiv \mathtt{x}((M \downarrow v)\langle w := O\rangle) \quad \text{(by } (gc)) \\ &\equiv \mathtt{x}(O) \quad \text{(by (x}var\text{))}. \\ RHS &\equiv \mathtt{x}(M\langle w \uparrow v := O \uparrow (v \downarrow (w \uparrow v))\rangle \\ &\qquad \langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle) \\ &\equiv \mathtt{x}(O \uparrow (v \downarrow (w \uparrow v))\langle v \downarrow (w \uparrow v) := \\ &\qquad N\langle w := O\rangle\rangle) \text{ (by (x}var\text{))} \\ &\equiv \mathtt{x}(O) \quad \text{(by } (gc)). \end{aligned}$$

Note that the (xvar) can be used above since $w \uparrow v \equiv M$ and (gc) is used since we have:
$(O \uparrow u)\langle u := Q\rangle \equiv O$,
which can easily be proved by induction on the construction of O.

(c) Otherwise ($M \not\equiv v$ and $M \downarrow v \not\equiv w$ i.e., $w \uparrow v \not\equiv M$).

$$\begin{aligned} LHS &\equiv \mathtt{x}(M\langle v := N\rangle\langle w := O\rangle) \\ &\equiv \mathtt{x}(M \downarrow v\langle w := O\rangle) \quad \text{(by } (gc)) \\ &\equiv \mathtt{x}((M \downarrow v) \downarrow w) \quad \text{(by } (gc)). \\ RHS &\equiv \mathtt{x}(M\langle w \uparrow v := O \uparrow (v \downarrow (w \uparrow v))\rangle \\ &\qquad \langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle) \\ &\equiv \mathtt{x}(M \downarrow (w \uparrow v)\langle v \downarrow (w \uparrow v) := \\ &\qquad N\langle w := O\rangle\rangle) \quad \text{(by } (gc)) \\ &\equiv \mathtt{x}(M \downarrow (w \uparrow v) \downarrow (v \downarrow (w \uparrow v))) \\ &\qquad \text{(by } (gc)) \\ &\equiv \mathtt{x}((M \downarrow v) \downarrow w). \end{aligned}$$

Note that, we can use the first (gc) above since $w \uparrow v \not\equiv M$, and the second since $M \not\equiv v$ and then $M \downarrow (w \uparrow v) \not\equiv v \downarrow (w \uparrow v)$. Also it can easily be verified that $M \downarrow (w \uparrow v) \downarrow (v \downarrow (w \uparrow v)) \equiv (M \downarrow v) \downarrow w$.

(2) $M \equiv \lambda z.Q$ where z is a pure variable.
It is desired to obtain:

$\mathtt{x}((\lambda z.Q)\langle v := N\rangle\langle w := O\rangle)$
$\equiv \mathtt{x}((\lambda z.Q)\langle w \uparrow v := O \uparrow (v \downarrow (w \uparrow v))\rangle$
$\langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle)$.
$\mathtt{x}((\lambda z.Q)\langle v := N\rangle\langle w := O\rangle)$
$\equiv \mathtt{x}((\lambda z.Q\langle v \uparrow z := N \uparrow z\rangle)\langle w := O\rangle)$
$\equiv \mathtt{x}(\lambda z.Q\langle v \uparrow z := N \uparrow z\rangle\langle w \uparrow z := O \uparrow z\rangle)$
$\equiv \lambda z.\mathtt{x}(Q\langle v \uparrow z := N \uparrow z\rangle\langle w \uparrow z := O \uparrow z\rangle)$
$\overset{\text{IH}}{\equiv} \lambda z.\mathtt{x}(Q\langle (w \uparrow z) \uparrow (v \uparrow z) := (O \uparrow z) \uparrow ((v \uparrow z) \downarrow ((w \uparrow z) \uparrow (v \uparrow z)))\rangle\langle (v \uparrow z) \downarrow ((w \uparrow z) \uparrow (v \uparrow z)) := (N \uparrow z)\langle w \uparrow z := O \uparrow z\rangle\rangle)$
$\equiv \mathtt{x}(\lambda z.Q\langle (w \uparrow z) \uparrow (v \uparrow z) := (O \uparrow z) \uparrow ((v \uparrow z) \downarrow ((w \uparrow z) \uparrow (v \uparrow z)))\rangle\langle (v \uparrow z) \downarrow ((w \uparrow z) \uparrow (v \uparrow z)) := (N\langle w := O\rangle) \uparrow z\rangle)$
$\equiv \mathtt{x}((\lambda z.Q\langle (w \uparrow z) \uparrow (v \uparrow z) := (O \uparrow z) \uparrow ((v \uparrow z) \downarrow (w \uparrow z) \uparrow (v \uparrow z))\rangle)\langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle)$
$\equiv \mathtt{x}((\lambda z.Q)\langle w \uparrow v := O \uparrow (v \downarrow (w \uparrow v))\rangle$
$\langle v \downarrow (w \uparrow v) := N\langle w := O\rangle\rangle)$.

Note that, in the above derivation we used the following equalities which can easily be verified when z is a pure variable:
$(N \uparrow z)\langle w \uparrow z := O \uparrow z\rangle \equiv (N\langle w := O\rangle) \uparrow z$.
$(v \uparrow z) \downarrow ((w \uparrow z) \uparrow (v \uparrow z)) \equiv (v \downarrow (w \uparrow v)) \uparrow z$.
$(w \uparrow z) \uparrow (v \uparrow z) \equiv (w \uparrow v) \uparrow z$.
$(O \uparrow z) \uparrow (v \uparrow z \downarrow (w \uparrow z) \uparrow (v \uparrow z)) \equiv O \uparrow (v \downarrow (w \uparrow v)) \uparrow z$.

(3) $M \equiv Q'Q''$.
Easy.
(4) $M \equiv \{\overline{x}\}.Q$.
Similar to case 2.
(5) $M \equiv Q \bullet [\overline{\langle x := Q'\rangle}]$.
Easy. □

Corollary 1 If M is an $\mathtt{x}$-normal term and y is a pure variable, then:
$\mathtt{x}(M\langle y := N\rangle\langle w := O\rangle) \equiv \mathtt{x}(M\langle w \uparrow y := O \uparrow y\rangle\langle y := N\langle w := O\rangle\rangle)$.

Proof This can easily be concluded from Lemma 4 by observing that $y \downarrow (w \uparrow y) = y$.

The following lemma will be used in the proof of Lemma 6.

Lemma 5 (General form of the substitution lemma) If M is an $\mathtt{x}$-normal λxc-term, then:
$\mathtt{x}(M\langle x_1 := N_1\rangle \dots \langle x_n := N_n\rangle\langle w := O\rangle) \equiv \mathtt{x}(M\langle w \uparrow \{\overline{x}\} := O \uparrow \{\overline{x}\}\rangle\langle x_1 := N_1\langle w \uparrow \overline{x_{n,2}} := O \uparrow \overline{x_{n,2}}\rangle\rangle \dots \langle x_n := N_n\langle w := O\rangle\rangle)$.

Proof By applying Corollary 1 repeatedly.

In proving Lemma 7 we need the following lemma.

Lemma 6 If $\mathtt{x}(M) \Rightarrow \mathtt{x}(M')$ and $\mathtt{x}(N) \Rightarrow \mathtt{x}(N')$, then $\mathtt{x}(M\langle v := N\rangle) \Rightarrow \mathtt{x}(M'\langle v := N'\rangle)$.

Proof By induction on the construction of $\mathtt{x}(M)$. Since $\mathtt{x}(M)$ is $\mathtt{x}$-normal, we have the following cases:

(1) $\mathtt{x}(M)$ is a variable.

(a) $\mathtt{x}(M) \equiv v$. We have $v \Rightarrow v \equiv \mathtt{x}(M')$.

$\mathtt{x}(M\langle v := N\rangle)$
$\equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)$
$\equiv \mathtt{x}(v\langle v := N\rangle)$
$\equiv \mathtt{x}(N)$
$\Rightarrow \mathtt{x}(N')$
$\equiv \mathtt{x}(v\langle v := N'\rangle)$
$\equiv \mathtt{x}(\mathtt{x}(M')\langle v := N'\rangle)$
$\equiv \mathtt{x}(M'\langle v := N'\rangle)$.

(b) $\mathtt{x}(M) \equiv w$ and $w \not\equiv v$.
We have $w \Rightarrow w \equiv \mathtt{x}(M')$.

$\mathtt{x}(M\langle v := N\rangle)$
$\equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)$
$\equiv \mathtt{x}(w\langle v := N\rangle)$
$\equiv \mathtt{x}(w \downarrow v)$
$\equiv w \downarrow v$
$\Rightarrow w \downarrow v$
$\equiv \mathtt{x}(w\langle v := N'\rangle)$
$\equiv \mathtt{x}(\mathtt{x}(M')\langle v := N'\rangle)$
$\equiv \mathtt{x}(M'\langle v := N'\rangle)$.

(2) $\mathtt{x}(M) \equiv \lambda x.P$, $P \Rightarrow P'$ and $\mathtt{x}(M') \equiv \lambda x.P'$.
From the induction hypothesis we have:
$\mathtt{x}(P\langle v \uparrow x := N \uparrow x\rangle) \Rightarrow \mathtt{x}(P'\langle v \uparrow x := N' \uparrow x\rangle)$

$\mathtt{x}(M\langle v := N\rangle)$
$\equiv \mathtt{x}((\lambda x.P)\langle v := N\rangle)$
$\equiv \mathtt{x}(\lambda x.P\langle v \uparrow x := N \uparrow x\rangle)$
$\equiv \lambda x.\mathtt{x}(P\langle v \uparrow x := N \uparrow x\rangle)$
$\Rightarrow \lambda x.\mathtt{x}(P'\langle v \uparrow x := N' \uparrow x\rangle)$
$\equiv \mathtt{x}(\lambda x.P'\langle v \uparrow x := N' \uparrow x\rangle)$
$\equiv \mathtt{x}((\lambda x.P')\langle v := N'\rangle)$.

(3) $\mathtt{x}(M) \equiv PQ$, $P \Rightarrow P'$, $Q \Rightarrow Q'$ and $\mathtt{x}(M') \equiv P'Q'$.
From the induction hypothesis we have:
$\mathtt{x}(P\langle v := N\rangle) \Rightarrow \mathtt{x}(P'\langle v := N'\rangle)$, and
$\mathtt{x}(Q\langle v := N\rangle) \Rightarrow \mathtt{x}(Q'\langle v := N'\rangle)$.

$\mathtt{x}(M\langle v := N\rangle)$
$\equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)$
$\equiv \mathtt{x}((PQ)\langle v := N\rangle)$
$\equiv \mathtt{x}(P\langle v := N\rangle)\mathtt{x}(Q\langle v := N\rangle)$
$\Rightarrow \mathtt{x}(P'\langle v := N'\rangle)\mathtt{x}(Q'\langle v := N'\rangle)$
$\equiv \mathtt{x}((P'Q')\langle v := N'\rangle)$
$\equiv (M'\langle v := N'\rangle)$.

(4) $\mathtt{x}(M) \equiv (\lambda y.P)Q$, $P \Rightarrow P'$, $Q \Rightarrow Q'$ and $\mathtt{x}(M') \equiv \mathtt{x}(P'\langle y := Q'\rangle)$.

From the induction hypothesis we have:
$\mathtt{x}(P\langle v \uparrow y := N \uparrow y\rangle) \Rightarrow$
$\quad \mathtt{x}(P'\langle v \uparrow y := N' \uparrow y\rangle)$, and
$\mathtt{x}(Q\langle v := N\rangle) \Rightarrow \mathtt{x}(Q'\langle v := N'\rangle)$.

$$\begin{aligned}
&\mathtt{x}(M\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(((\lambda y.P)Q)\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}((\lambda y.P)\langle v := N\rangle)\mathtt{x}(Q\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(\lambda y.P\langle v \uparrow y := N \uparrow y\rangle)\\
&\qquad \mathtt{x}(Q\langle v := N\rangle)\\
&\quad \Rightarrow \mathtt{x}(\mathtt{x}(P'\langle v \uparrow y := N' \uparrow y\rangle)\\
&\qquad \langle y := \mathtt{x}(Q'\langle v := N'\rangle)\rangle)\\
&\quad \equiv \mathtt{x}(P'\langle v \uparrow y := N' \uparrow y\rangle\\
&\qquad \langle y := Q'\langle v := N'\rangle\rangle)\\
&\quad \overset{\text{Cor.1}}{\equiv} \mathtt{x}(P'\langle y := Q'\rangle\langle v := N'\rangle)\\
&\quad \equiv \mathtt{x}(M'\langle v := N'\rangle).
\end{aligned}$$

(5) $\mathtt{x}(M) \equiv \{\overline{x}\}.P$, $P \Rightarrow P'$, and $\mathtt{x}(M') \equiv \{\overline{x}\}.P'$.

$$\begin{aligned}
&\mathtt{x}(M\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}((\{\overline{x}\}.P)\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(\{\overline{x}\}.P\langle v \uparrow \{\overline{x}\} := N \uparrow \{\overline{x}\}\rangle)\\
&\quad \equiv \{\overline{x}\}.\mathtt{x}(P\langle v \uparrow \{\overline{x}\} := N \uparrow \{\overline{x}\}\rangle)\\
&\quad \Rightarrow \{\overline{x}\}.\mathtt{x}(P'\langle v \uparrow \{\overline{x}\} := N' \uparrow \{\overline{x}\}\rangle)\\
&\quad \equiv \mathtt{x}(\{\overline{x}\}.P'\langle v \uparrow \{\overline{x}\} := N' \uparrow \{\overline{x}\}\rangle)\\
&\quad \equiv \mathtt{x}((\{\overline{x}\}.P')\langle v := N'\rangle)\\
&\quad \equiv \mathtt{x}(N'\langle v := N'\rangle).
\end{aligned}$$

(6) $\mathtt{x}(M) \equiv P \bullet [\langle x_1 := Q_1\rangle \ldots \langle x_n := Q_n\rangle]$, $P \Rightarrow P'$, $Q_i \Rightarrow Q'_i$, $(1 \le i \le n)$, and $\mathtt{x}(M') \equiv P' \bullet [\langle x_1 := Q'_1\rangle \ldots \langle x_n := Q'_n\rangle]$.

$$\begin{aligned}
&\mathtt{x}(M\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}((P \bullet [\langle x_1 := Q_1\rangle \ldots \langle x_n := Q_n\rangle\\
&\qquad])\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(P\langle v := N\rangle \bullet [\langle x_1 := Q_1\\
&\qquad \langle v \uparrow \overline{x_{n,2}} := N \uparrow \overline{x_{n,2}}\rangle\rangle\\
&\qquad \cdots \langle x_n := Q_n\langle v := N\rangle\rangle])\\
&\quad \Rightarrow \mathtt{x}(P'\langle v := N'\rangle) \bullet [\langle x_1 :=\\
&\qquad \mathtt{x}(Q'_1\langle v \uparrow \overline{x_{n,2}} := N' \uparrow \overline{x_{n,2}}\rangle)\rangle\\
&\qquad \cdots \langle x_n := \mathtt{x}(Q'_n\langle v := N'\rangle)\rangle]\\
&\quad \equiv \mathtt{x}(P'\langle v := N'\rangle \bullet [\langle x_1 := Q'_1\\
&\qquad \langle v \uparrow \overline{x_{n,2}} := N' \uparrow \overline{x_{n,2}}\rangle\rangle \cdots\\
&\qquad \langle x_n := Q'_n\langle v := N'\rangle\rangle])\\
&\quad \equiv \mathtt{x}((P' \bullet [\langle x_1 := Q'_1\rangle \ldots \langle x_n := Q'_n\rangle\\
&\qquad])\langle v := N'\rangle)\\
&\quad \equiv \mathtt{x}(M'\langle v := N'\rangle).
\end{aligned}$$

(7) $\mathtt{x}(M) \equiv (\{\overline{x}\}.P) \bullet [\langle x_1 := Q_1\rangle \ldots \langle x_n := Q_n\rangle]$, $P \Rightarrow P'$, $Q_i \Rightarrow Q'_i$, $(1 \le i \le n)$, and $\mathtt{x}(M') \equiv \mathtt{x}(P'\langle x_1 := Q'_1\rangle \ldots \langle x_n := Q'_n\rangle)$.
From the induction hypothesis we have:

$$\begin{aligned}
&\mathtt{x}(P\langle v \uparrow \{\overline{x}\} := N \uparrow \{\overline{x}\}\rangle) \Rightarrow\\
&\qquad \mathtt{x}(P'\langle v \uparrow \{\overline{x}\} := N' \uparrow \{\overline{x}\}\rangle),\\
&\mathtt{x}(Q_1\langle v \uparrow \overline{x_{n,2}} := N \uparrow \overline{x_{n,2}}\rangle) \Rightarrow\\
&\qquad \mathtt{x}(Q'_1\langle v \uparrow \overline{x_{n,2}} := N' \uparrow \overline{x_{n,2}}\rangle),\\
&\qquad \vdots\\
&\mathtt{x}(Q_n\langle v := N\rangle) \Rightarrow \mathtt{x}(Q'_n\langle v := N'\rangle).
\end{aligned}$$

$$\begin{aligned}
&\mathtt{x}(M\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(\mathtt{x}(M)\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}(((\{\overline{x}\}.P) \bullet [\langle x_1 := Q_1\rangle\\
&\qquad \cdots\\
&\qquad \langle x_n := Q_n\rangle])\langle v := N\rangle)\\
&\quad \equiv \mathtt{x}((\{\overline{x}\}.P)\langle v := N\rangle \bullet [\langle x_1 :=\\
&\qquad Q_1\langle v \uparrow \overline{x_{n,2}} := N \uparrow \overline{x_{n,2}}\rangle\rangle\\
&\qquad \cdots\\
&\qquad \langle x_n := Q_n\langle v := N\rangle\rangle])\\
&\quad \Rightarrow \mathtt{x}(\mathtt{x}(P'\langle v \uparrow \{\overline{x}\} := N' \uparrow \{\overline{x}\}\rangle)\\
&\qquad \langle x_1 := \mathtt{x}(Q'_1\langle v \uparrow \overline{x_{n,2}} := N' \uparrow\\
&\qquad \overline{x_{n,2}}\rangle)\rangle \cdots\\
&\qquad \langle x_n := \mathtt{x}(Q'_n\langle v := N'\rangle)\rangle)\\
&\quad \equiv \mathtt{x}(P'\langle v \uparrow \{\overline{x}\} := N' \uparrow \{\overline{x}\}\rangle\\
&\qquad \langle x_1 := Q'_1\langle v \uparrow \overline{x_{n,2}} := N' \uparrow \overline{x_{n,2}}\rangle\rangle\\
&\qquad \cdots \langle x_n := Q'_n\langle v := N'\rangle\rangle)\\
&\quad \equiv \mathtt{x}(P'\langle x_1 := Q'_1\rangle \ldots \langle x_n := Q'_n\rangle\\
&\qquad \langle v := N'\rangle) \quad \text{(by Lemma 5)}\\
&\quad \equiv \mathtt{x}(M'\langle v := N'\rangle). \qquad \Box
\end{aligned}$$

The following two lemmas are important in proving the confluence property of λxc.

Lemma 7 If $M \to_{b,c} M'$, then $\mathtt{x}(M) \Rightarrow \mathtt{x}(M')$.

Proof By induction on the construction of M:

(1) $M \equiv (\lambda x.N)Q$, $M' \equiv N\langle x := Q\rangle$ and $M \to_{b,c} M'$. We have:

$$\begin{aligned}
\mathtt{x}(M) &\equiv \mathtt{x}(\lambda x.N)(\mathtt{x}(Q))\\
&\equiv (\lambda x.\mathtt{x}(N))(\mathtt{x}(Q))\\
&\Rightarrow \mathtt{x}(\mathtt{x}(N)\langle x := \mathtt{x}(Q)\rangle)\\
&\equiv \mathtt{x}(N\langle x := Q\rangle)\\
&\equiv \mathtt{x}(M').
\end{aligned}$$

(2) $M \equiv (\{\overline{x}\}.P) \bullet [\overline{\langle x := N\rangle}]$, $M' \equiv P\overline{\langle x := N\rangle}$ and $M \to_{b,c} M'$. We have:

$$\begin{aligned}
\mathtt{x}(M) &\equiv \{\overline{x}\}.\mathtt{x}(P) \bullet [\overline{\langle x := \mathtt{x}(N)\rangle}]\\
&\Rightarrow \mathtt{x}(\mathtt{x}(P)\overline{\langle x := \mathtt{x}(N)\rangle})\\
&\equiv \mathtt{x}(P\overline{\langle x := N\rangle})
\end{aligned}$$

$\equiv \quad \mathtt{x}(M')$.

(3) $M \equiv \lambda x.N$, $N \rightarrow_{b,c} N'$ and $M' \equiv \lambda x.N'$. From the induction hypothesis we have: $\mathtt{x}(N) \Rightarrow \mathtt{x}(N')$. Hence, $\mathtt{x}(M) \equiv \lambda x.\mathtt{x}(N) \Rightarrow \lambda x.\mathtt{x}(N') \equiv \mathtt{x}(M')$.

(4) $M \equiv NQ$, $N \rightarrow_{b,c} N'$ and $M' \equiv N'Q$. From the induction hypothesis we have $\mathtt{x}(N) \Rightarrow \mathtt{x}(N')$. So, we have:

$$\begin{array}{rcl} \mathtt{x}(M) & \equiv & \mathtt{x}(NQ) \\ & \equiv & \mathtt{x}(N)\mathtt{x}(Q) \\ & \Rightarrow & \mathtt{x}(N')\mathtt{x}(Q) \\ & \equiv & \mathtt{x}(N'Q) \\ & \equiv & \mathtt{x}(M'). \end{array}$$

(5) $M \equiv NQ$, $Q \rightarrow_{b,c} Q'$ and $M' \equiv NQ'$: similar to the above case.

(6) $M \equiv N\langle v := Q\rangle$, $N \rightarrow_{b,c} N'$ and $M' \equiv N'\langle v := Q\rangle$. From the induction hypothesis we have:
$\mathtt{x}(N) \Rightarrow \mathtt{x}(N')$ and from the reflexivity of $\Rightarrow$ we have $\mathtt{x}(Q) \Rightarrow \mathtt{x}(Q)$.
From Lemma 6 we have $\mathtt{x}(N\langle v := Q\rangle) \Rightarrow \mathtt{x}(N'\langle v := Q\rangle)$.

(7) $M \equiv N\langle v := Q\rangle$, $Q \rightarrow_{b,c} Q'$ and $M' \equiv N\langle v := Q'\rangle$: similar to the above case.

(8) $M \equiv \{\overline{x}\}.N$, $N \rightarrow_{b,c} N'$ and $M' \equiv \{\overline{x}\}.N'$. From the induction hypothesis we have $\mathtt{x}(N) \Rightarrow \mathtt{x}(N')$, hence $\mathtt{x}(M) \equiv \{\overline{x}\}.\mathtt{x}(N) \Rightarrow \{\overline{x}\}.\mathtt{x}(N') \equiv \mathtt{x}(M')$.

(9) $M \equiv P \bullet [\overline{\langle x := N\rangle}]$, $P \rightarrow_{b,c} P'$ and $M' \equiv P' \bullet [\overline{\langle x := N\rangle}]$. From the induction hypothesis we have: $\mathtt{x}(P) \Rightarrow \mathtt{x}(P')$, hence

$$\begin{array}{rcl} \mathtt{x}(M) & \equiv & \mathtt{x}(P) \bullet [\overline{\langle x := \mathtt{x}(N)\rangle}] \\ & \Rightarrow & \mathtt{x}(P') \bullet [\overline{\langle x := \mathtt{x}(N)\rangle}] \\ & \equiv & \mathtt{x}(M'). \end{array}$$

(10) $M \equiv P \bullet [\overline{\langle x := N\rangle}]$, $N_i \rightarrow_{b,c} N_i'$, $i := 1,\ldots,n$ and $M' \equiv P \bullet [\overline{\langle x := N'\rangle}]$. From the induction hypothesis we have: $\mathtt{x}(N_i) \Rightarrow \mathtt{x}(N_i')$, hence

$$\begin{array}{rcl} \mathtt{x}(M) & \equiv & \mathtt{x}(P) \bullet [\overline{\langle x := \mathtt{x}(N)\rangle}] \\ & \Rightarrow & \mathtt{x}(P) \bullet [\overline{\langle x := \mathtt{x}(N')\rangle}] \\ & \equiv & \mathtt{x}(M'). \end{array}$$ □

Lemma 8 If $M \rightarrow_{\mathrm{x}} M'$ then $\mathtt{x}(M) \Rightarrow \mathtt{x}(M')$.

Proof Easy.

Remark 1 From Lemmas 7 and 8 we have, if $M \rightarrow_{\mathrm{bxc}} M'$, then $\mathtt{x}(M) \Rightarrow \mathtt{x}(M')$.

To prove the confluence of $\Rightarrow$, the following lemma is important.

Lemma 9 If $M \Rightarrow N$, then $N \Rightarrow M^*$.

Proof By induction on the construction of M.

Lemma 10 $\Rightarrow$ on x-normal terms is confluent.

Proof Immediate consequence of Lemma 9.

Theorem 1 (Confluence) $\rightarrow_{\mathrm{bxc}}$ on λxc-terms is confluent.

Proof Suppose that $M \rightarrow^*_{\mathrm{bxc}} N$ and $M \rightarrow^*_{\mathrm{bxc}} P$, then from Remark 1 we have $\mathtt{x}(M) \Rightarrow \mathtt{x}(N)$ and $\mathtt{x}(M) \Rightarrow \mathtt{x}(P)$, and from the confluency of $\Rightarrow$ (Lemma 10) there is Q s.t. $\mathtt{x}(N) \Rightarrow Q$ and $\mathtt{x}(P) \Rightarrow Q$. Then from Lemma 2 we have $\mathtt{x}(N) \rightarrow^*_{\mathrm{bxc}} Q$ and $\mathtt{x}(P) \rightarrow^*_{\mathrm{bxc}} Q$. Since $N \rightarrow^*_{\mathrm{bxc}} \mathtt{x}(N)$ and $P \rightarrow^*_{\mathrm{bxc}} \mathtt{x}(P)$, we have $N \rightarrow^*_{\mathrm{bxc}} Q$ and $P \rightarrow^*_{\mathrm{bxc}} Q$. □

3.2 Conservativity

In proving that λxc is a conservative extension of the $\lambda\beta$ calculus, we need the following definitions.

Two terms M and N are *α-equivalent*, written as $M \equiv_\alpha N$, if they are identical except for renaming of bound variables bound by λ and by the v in the term $P\langle v := Q\rangle$ only, and is defined inductively as:

(1) $v \equiv_\alpha v$.

(2) $\lambda x.M \equiv_\alpha \lambda y.N$ if $(M[x := z]) \equiv_\alpha (N[y := z])$ for some $z \notin FV(MN)$.

(3) $MN \equiv_\alpha PQ$ if $M \equiv_\alpha P$ and $N \equiv_\alpha Q$.

(4) $M\langle v := N\rangle \equiv_\alpha P\langle w := Q\rangle$ if $N \equiv_\alpha Q$ and $M[v := u] \equiv_\alpha P[w := u]$ for some $u \notin FV(MP)$.

(5) $\{\overline{x}\}.M \equiv_\alpha \{\overline{x}\}.N$ if $M \equiv_\alpha N$.

(6) $M \bullet [\overline{\langle x := N\rangle}] \equiv_\alpha P \bullet [\overline{\langle x := Q\rangle}]$ if $M \equiv_\alpha P$ and $N_i \equiv_\alpha Q_i$, for $1 \leq i \leq n$.

Example 5 $\lambda z.z + x \equiv_\alpha \lambda x.x + \#x$.

Note that, if $M \equiv_\alpha N$ then $\mathtt{FV}(M) = \mathtt{FV}(N)$.

The notation $P[v := N]$ denotes the term obtained by substitution of the term N for all free occurrences of v in P and is defined inductively as:

(1) $v[v := N] \equiv N$.

(2) $w[v := N] \equiv w \downarrow v$ if $\quad v \neq w$.

(3) $(\lambda y.M)[v := N] \equiv \lambda y.M[v \uparrow y := N \uparrow y]$.

(4) $(M_1M_2)[v := N] \equiv (M_1[v := N])(M_2[v := N])$.

(5) $(P\langle w := M\rangle)[v := N] \equiv P[v \uparrow w := N \uparrow (w \downarrow (v \uparrow w))]\langle w \downarrow (v \uparrow w) := M[v := N]\rangle$.

(6) $(\{\overline{x}\}.M)[v := N] \equiv \{\overline{x}\}.M[v \uparrow \{\overline{x}\} := N \uparrow \{\overline{x}\}]$.

(7) $(M \bullet [\overline{\langle x := P\rangle}])[v := N]$
$\equiv M[v := N] \bullet$
$[\langle x_1 := P_1[v \uparrow \overline{x_{n,2}} := N \uparrow \overline{x_{n,2}}]\rangle$

$\cdots$
$\langle x_n := P_n[v := N]\rangle]$.

Theorem 2 (The Reduction in λxc respects the α-equivalence) If $M \equiv_\alpha N$ and $M \rightarrow_{\mathrm{bxc}} M'$ then there is a term N' s.t. $N \rightarrow_{\mathrm{bxc}} N'$ and $M' \equiv_\alpha N'$.

Proof By induction on the construction of M.

Note that, the calculus defined by Sato, et al.[8] cannot express the above theorem as it is given here, since M' and N' do not always have the same type and then they are not necessarily α-equivalent.

If M and N are λxc-terms with only variables, λ-abstraction and application, then they can be regarded as λβ-terms.

Theorem 3 (Conservativity) If M and N are λβ-terms, then $M \stackrel{*}{\rightarrow}_\beta N$ iff $M \rightarrow^*_{\mathrm{bxc}} N'$ and $N \equiv_\alpha N'$ for some term N' in λxc.

Proof By induction on the construction of M.

3.3 PSN

Finally, we will show that λxc has the PSN property, which states that if a λβ-term M is strongly normalizing under the ordinary β-reduction, then it is also strongly normalizing under λxc-reductions. We will follow the method given in Ref. 2). Another method can be found in Ref. 1).

First, a garbage-free reduction $\rightarrow_{\mathrm{bxc}|gc}$ is defined as follows:

Let M and N be gc-normal terms (M is gc-normal if $M \rightarrow_{gc} P$ holds for no P) then $M \rightarrow_{\mathrm{bxc}|gc} N$ iff $\exists\, P$ s.t.

$$M \rightarrow_{\mathrm{x}} P \text{ and } P \stackrel{*}{\rightarrow}_{gc} N \text{ or}$$
$$M \rightarrow_{\mathrm{b}} P \text{ and } P \stackrel{*}{\rightarrow}_{gc} N \text{ or}$$
$$M \rightarrow_{\mathrm{C}} P \text{ and } P \stackrel{*}{\rightarrow}_{gc} N.$$

The garbage-free reduction calculus will be denoted by λxc$|gc$, and the gc-normal form of M will be denoted by $gc(M)$.

Theorem 4 (PSN for λxc$|gc$) λβ-terms that are β-strongly normalizing are also strongly normalizing for λxc$|gc$.

Proof Since λβ-terms has no reduction with the (c) rule, the proof is a straight forward extension of that of $\lambda\mathtt{x}$[2].

Next, *garbage-reduction* is defined as the contextual closure of the reduction generated by:

(1) If $N \rightarrow_{\mathrm{bxc}} N'$ and $v \notin \mathtt{FV}(gc(M))$, then $M\langle v := N\rangle \rightarrow_{\mathrm{bxc}} M\langle v := N'\rangle$ is a garbage-reduction.

(2) If $v \notin \mathtt{FV}(gc(MN))$ then $(MN)\langle v := P\rangle \rightarrow_{\mathrm{bxc}} (M\langle v := P\rangle)\ (N\langle v := P\rangle)$ is a garbage-reduction.

(3) If $v \notin \mathtt{FV}(gc(\lambda y.M))$ then $(\lambda y.M)\langle v := N\rangle \rightarrow_{\mathrm{bxc}} \lambda y.M\langle v \uparrow y := N \uparrow y\rangle$ is a garbage-reduction.

(4) If $v \notin \mathtt{FV}(M)$ then $M\langle v := N\rangle \rightarrow_{\mathrm{bxc}} M \downarrow v$ is a garbage-reduction.

(5) If $v \notin \mathtt{FV}(gc(\{\overline{y}\}.M))$ then $(\{\overline{y}\}.M)\langle v := P\rangle \rightarrow_{\mathrm{bxc}} \{\overline{y}\}.M\langle v \uparrow \{\overline{y}\} := P \uparrow \{\overline{y}\}\rangle$ is a garbage-reduction.

(6) If $v \notin \mathtt{FV}(gc(M \bullet [\overline{\langle y := P\rangle}]))$ then $(M \bullet [\overline{\langle y := P\rangle}])\langle v := N\rangle \rightarrow_{\mathrm{bxc}} M\langle v := N\rangle \bullet [\langle y_1 := P_1\langle v \uparrow \overline{y_{n,2}} := N \uparrow \overline{y_{n,2}}\rangle\rangle \ldots \langle y_n := P_n\langle v := N\rangle\rangle]$ is a garbage-reduction.

The following proposition and lemmas are needed in the proof of Theorem 5.

Proposition 1 If $M \rightarrow_{\mathrm{bxc}} N$ is not a garbage-reduction then $gc(M) \rightarrow_{\mathrm{bxc}|gc} gc(N)$

Proof By induction on the construction of M.

Recall the following definition from Ref. 2): N is said to be body of a substitution in M if for some P and x we have $P\langle x := N\rangle$ is a subterm of M. The predicate $subSN(M)$ should be read to be all bodies of substitutions in M are strongly normalizing for λxc-reduction.

Lemma 11 Let M be a λxc-term, then if $subSN(M)$ and $M \rightarrow_{\mathrm{bxc}} N$ is a garbage-reduction then $subSN(N)$.

Proof Easy.

Lemma 12 If $subSN(M)$ then M is strongly normalizing for garbage-reduction.

Proof The proof is a straightforward extension of that of λx[2].

For all terms M, define $Ngf(M)$ to be the maximum length of garbage-free reduction paths starting in $gc(M)$.

Theorem 5

If $Ngf(M) < \infty$ and $subSN(M)$ then M is strongly normalizing for $\rightarrow_{\mathrm{bxc}}$-reduction.

Proof By induction on $Ngf(M)$ using Lemmas 11, 12 and Proposition 1.

Corollary 2 (PSN for λxc) A λβ-term is strongly normalizing for β-reduction iff it is strongly normalizing for λxc-reduction.

Proof Using Theorems 4 and 5, Lemma 12, and the fact that for λ_β-term M if $M \rightarrow_\beta N$ then $M \stackrel{*}{\rightarrow}_{b,\mathtt{x}} \mathtt{x}(N)$.

4. Conclusion

We have developed a type free calculus for contexts, which is an extension of the explicit substitution calculus λx. In this calculus contexts and lambda terms share the same set of variables and can be freely mixed. Also, contexts are first-class values and hole-filling is an explicit operation.

We have shown that λxc is confluent, conservative over type free $\lambda\beta$ calculus and has the PSN property, i.e., pure in the sense of Ref. 9).

Unlike the system defined in Ref. 8), we restrict the variables bound by the set $\{\overline{x}\}$ as well as the variables bound by λ to pure variables, which makes our calculus simpler and it does not affect our intended motivation about contexts. Our motivation behind giving the variables one or more # is to avoid collision with other existing variables. However, in the process of writing programs we can choose these variables (the variables bound by λ and the variables bound by the set $\{\overline{x}\}$) as pure variables, and during the computation according to the λxc-rules they will never take #.

For future work, we suggest defining a set of typing rules for the terms of λxc to get a typed version of this calculus which includes part of Martin-Löf's type theory ML_0, e.g., sum, product, well-ordering, etc. For the resulting typing calculus, designing a type inference algorithm which produces principal typing for each typable term is also promising.

References

1) Bloo, R. and Geuvers, H.: Explicit Substitution on the edge of Strong Normalization, *Theoretical Computer Science*, Vol.211, pp.375–395 (1999).
2) Bloo, R. and Rose, K.H.: Preservation of Strong Normalization in Named Lambda Calculi with Explicit Substitution and Garbage Collection, van Vliet, J.C. (Ed.), *Proc. Computer Science in Netherlands '95* (1995). (`ftp://ftp.diku.dk/diku/semantics/papers/D-246.ps`)
3) Dami, L.: A Lambda-Calculus for Dynamic Binding, *Theoretical Computer Science*, Vol.192, pp.201–231 (1998).
4) Hashimoto, M. and Ohori, A.: A Typed Context Calculus, *Theoretical Computer Science*, (to appear).
5) Klop, J.W., et al.: Combinatory reduction systems: Introduction and Survey, *Theoretical Computer Science*, Vol.121, pp.279–308 (1993).
6) Mason, I.: Computing with Contexts, *Higher-Order and Symbolic Computation*, Vol.12, No.2, pp.171–201 (1999).
7) Sands, D.: Computing with Contexts – A Simple Approach, *Electronic Notes in Theoretical Computer Science* (1998).
8) Sato, M., Sakurai, T. and Kameyama, Y.: A Simply Typed Context Calculus with First Class Environment, *Proc. Fifth International Symposium on Functional and Logic Programming*, Lecture Notes in Computer Science (to appear).
9) Sato, M., Sakurai, T. and Burstall, R.: Explicit Environments, *Lecture Notes in Computer Science*, Vol.1581, pp.340–354 (1999).
10) Takahashi, M.: Parallel Reduction in λ-calculus, *J. Symbolic Computation*, Vol.7, pp.113–123 (1989).

(Received November 25, 1999)
(Accepted October 6, 2000)

Azza A. Taha received the Master of Science degree in Computer Science from Ain Shams University, Cairo, Egypt. She is a Ph.D. student at Kyoto University since April 1997. Her research interests are in foundation of information science, lambda-calculus and type theory.

Masahiko Sato received the degree of Master of Science in Mathematics from University of Tokyo in 1973, and the degree of Doctor of Science in Mathematics from Kyoto University in 1977. He is currently a professor in the Graduate School of Informatics, Kyoto University. His main research interest is theory of programs, especially constructive programming. He is a member of IPSJ, JSSST, and Mathematical Society of Japan.

Yukiyoshi Kameyama received the degrees of Master of Science from University of Tokyo and Doctor of Engineering from Kyoto University. He belongs to the Graduate School of Informatics, Kyoto University. His main research interests are in foundation of software science, including logic in computation, type theory, and functional programming. He is a member of JSSST and IPSJ.