Theoretical Computer Science 433 (2012) 1–19

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Alpha equivalence equalities

Roy L. Crole *

Department of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, UK

ARTICLE INFO

Article history:
Received 8 November 2010
Received in revised form 27 September 2011
Accepted 15 January 2012
Communicated by D. Sannella

Keywords:
α-equivalence
Atom
Context
λ-expression
Permutation action
Renaming
Variable binding

ABSTRACT

Programming languages and logics, which are pervasive in Computer Science, have syntax which involves variable binding constructors. As such, reasoning about such languages in general, and formal reasoning in particular (such as within a theorem prover), requires frameworks within which the syntax may be properly represented. One key requirement is a correct representation of α-equivalence.

The current literature provides a number of different definitions of the notion of α-equivalence. The formal definitions may be nameless as in the approach of de Bruijn, or have explicit names, as in the approaches that use either a renaming/substitution axiom, or instead use a notion of variable swapping.

The first contribution of this paper is to draw together five definitions of α-equivalence relations and to prove *formally and in detail, but using mathematics,* that the relations are all equal. There are two key reasons for doing this: Firstly, the literature has many examples of proofs of results involving α-equivalence which contain *technical* errors. Such examples concern both the application of α-equivalence, and the meta-theory of α-equivalence itself. Secondly, the literature does not currently contain detailed presentations of such results. The point of giving the detail is partly to avoid falling into common error-traps, but mainly to provide clear mathematical machinery that will be useful to those working in the area. This includes systems of inductive rules and proofs by induction, and clear accounts of the key lemmas that support the main proofs.

The second contribution is to provide two definitions of α-equivalence relations over *(program) contexts*, namely expressions with a single meta-variable (or "hole"). One of the definitions is already in the literature, and the other is new. We prove some basic properties of α-equivalence on contexts, and show that the two definitions give rise to the same relation.

1. Introduction

The research community, and even many students, are well aware that de Bruijn λ-expressions, and what one might term "traditional" α-equivalence classes of λ-expressions with binding names, are "the same" (here we are thinking of equivalence classes $[\lambda\, v.\, E]_\alpha$ generated by rules including a renaming axiom $\lambda\, v.\, E \sim \lambda\, v'.\, E\{v'/v\}$ defined using capture-avoiding substitution). In previous work [5] (Section 4.1, Theorem 4.1), the author formally specified a bijection between *locally nameless* de Bruijn λ-expressions and α-equivalence classes of λ-expressions with binding names. It is not *too* difficult to specify the pair of inverse functions formally, but *it is* surprisingly awkward to give a proof that they are indeed mutually inverse and certainly *to get the proof correct*. For a similar result that is actually mechanized see [21] (indeed a summary of various possible bijections appears at the start of Section 2, page 208).

* Tel.: +44 116 252 3404; fax: +44 116 252 3604.
E-mail address: r.crole@mcs.le.ac.uk.

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.01.030

On various occasions the author has been asked questions about the many other definitions of α-equivalence (with names) that are now available, especially those given using a notion of name swapping in [12] (Theorem 2.1, page 216), or [13] (Section 2, page 3). It became clear through whiteboard discussions that proving that the many definitions are all the same is quite involved—certainly more so than one might first expect.

In the literature one finds many formal errors in proofs of results of this nature. For example, Hindley and Seldin [16] (Section 1B, page 10) write "The above technicalities are rather dull. But their very dullness has made them a trap for even the most careful authors". Indeed one can find errors within proofs in [14] (page 91), one of the major works by Curry and Feys. In the papers [12] (all of Section 2; including Theorem 2.1, page 216) and [13] (Section 2, page 3) that introduced name swapping, the (very short) proof outline that the new definition of α-equivalence via swapping is the same as one defined through the axiom of α-equivalence, is also not quite right—see page 4 of our article.

Even formal proofs of simple results that depend upon α-equivalence classes of expressions, such as the weakening of contexts of variables, present hidden traps. McKinna and Pollack [20] (Sections 5.2 and 5.3, page 29–31) consider the α-equivalence classes of expressions of λ-calculus in contexts of typed variables. In trying to prove that contexts may be weakened (Section 5.3), they observe that if one considers all of the possible cases in a standard induction proof, then one "encounters serious difficulties with parameter [atom] side conditions". They alleviate the problem by defining an alternative typing relation (Section 5.2, page 29) "that identifies all those derivations of each judgment that are inessentially different because of parameters occurring in the derivation but not in its conclusion" (as is the case with rules dealing with binding). The alternative typing relation has premises required to hold for all "sufficiently fresh" parameters. See [24] for general discussions of these issues.

As explained in [23] (Section 2, page 170), one may also deal with weakening by showing that the rules of inference used to define the expressions in context are invariant under name-swapping (c.f. "inessentially different because of parameters"), and from this one may complete a formal proof. The point here, once again, is that errors (or at least significant omissions) may creep in if one does not properly consider all the details.

In [10] (Section 5, page 21) Gabbay considers a bijection between two definitions of α-equivalence classes. One is specified using atom abstraction, and the other makes use of capture avoiding substitution. Although the definitions are different to those found in our paper, it is worth noting that there is commonality in some key lemmas—see Section 6.1.

The primary purpose of this paper is to draw together five formulations of α-equivalence and to show they are all the same. Three are based on name swapping and two are based on the traditional α-equivalence axiom (renaming to avoid capture). While it would be pointless to provide every single line of the proofs, we do give considerable detail. The point of doing this is to try to ensure that everything *really* is correct—and to avoid the kinds of pitfalls that a number of other authors have fallen into. In the main, I have chosen to give more details of the proofs involving swapping, and less for those involving renaming. Name swapping is a newer idea, and the literature has far fewer detailed name-swapping proofs than of those involving renaming. Since it would not be possible (at least, certainly not desirable) to give every single detail of every proof, the reader might reasonably wonder if there are still any errors. Of course, there may be. The intention has been to give *enough* detail to make sure that all required lemmas are present, and that, with a little additional work the reader might be able to fill in all the gaps if required. We also intend our paper to be one centering on clear mathematical proofs: the reader might also ask about mechanical proofs, and we take up the idea in Section 7.

In addition we also consider two formulations of α-equivalence on program contexts. Traditionally a program context is an expression with a single hole, or, more precisely, a single meta-variable. Our first definition arises as a special case of a definition in [26] (Figures 1 and 2, page 480). Our second is a new definition which appears to be the analogue, for program contexts, of Definition 3.1. We prove that the two formulations give rise to the same α-equivalence relation.

In the remainder of the paper, we use a notion of a set of *atoms* $\mathbb{A}$ to act as a set of names. This means that the set is countable and the only operation that may be performed is a test for equality of the names of a pair of atoms.

2. Fundamentals of atom swapping

This paper is based on a set of simple expressions that are built out of a single non-binding binary-expression constructor P, and a single unary-binding unary-expression constructor B. We choose the notation below to emphasize that the work applies to binding in general, with nothing specific to λ-expressions—however the heart of each proof still remains valid if $\mathsf{P}(E_1, E_2)$ is replaced with E_1E_2 and $\mathsf{B}([a]E)$ is replaced with $\lambda\, a.\, E$. Thus we consider a set of **syntactic expressions** $E \in \mathit{Exp}$ defined by the following grammar

$$E ::= a \mid \mathsf{P}(E, E) \mid \mathsf{B}([a]E)$$

where a is a meta-variable ranging over $\mathbb{A}$. We shall view an expression as a finite tree where nodes are constructors and leaves are atoms. In any subexpression $\mathsf{B}([a]E)$ the occurrence of a is a **binding** atom with **scope** E. All atoms a in scope, plus the binding atom, are **bound** occurrences. We say that the occurrences are **captured** by the binding atom. The definitions are of course standard (see for example [16], Section 1B, pages 5, 6) and of course the sets of occurring atoms $\mathit{occ}(E)$, free atoms $\mathit{free}(E)$ and bound atoms $\mathit{bnd}(E)$ may be defined by recursion on the structure of E.

If expressions E and E' are syntactically identical we write $E \equiv E'$. We shall also make use of the following notation

$a \lhd E$	a has at least one occurrence in E, i.e. $a \in occ(E)$
$a \not\lhd E$	a does not have any occurrences in E, i.e. $a \notin occ(E)$
$a \lhd_{\mathsf{fr}} E$	a has at least one free occurrence in E, i.e. $a \in free(E)$
$a \# E$	a does not have any free occurrences in E, i.e. $a \notin free(E)$

When using the (standard) notation $a \# E$ to mean that $a \notin free(E)$ we say that a is **fresh** for E.

We will make use of the **simultaneous capture avoiding substitution of atoms for free atoms**: given finite lists of atoms $L = \vec{a}$ and $L' = \vec{a'}$ and expression E, then the expression $E\{L'/L\}$ is E with all free occurrences of any atom, that occurs in L at position i, replaced by a' in position i in L', with renaming to avoid any captures taking place. Note that one might refer to such "substitution" of atoms for atoms as *renaming*. Since other functions, such as swapping, might also be regarded as "renaming" atoms, we have decided to stick with the term *substitution* throughout, to distinguish renamings arising from such capture avoiding substitution from other renamings. The formal definition of $E\{L'/L\}$ appears in the Appendix on page 18, together with a brief justification for its existence.

Remark Please note that while we seek to provide very full details of the proofs of theorems, it remains a central goal to give a mathematical presentation such that we retain a feel for the overall structure of proofs. As such, *we do not call upon the formal definition of atom substitution, just the existence of such a function*—for related results that do utilize the formal definition, and the consequent increase in detail, please see [5] (Appendix C, page 628). We talk about the author's current work on mechanized proofs in Section 7, which do require the material in the Appendix.

As in the papers [12,13] and thesis [3] we shall make heavy use of permutations π on $\mathbb{A}$ with finite support, that is, the **domain** of π

$$dom(\pi) \stackrel{\text{def}}{=} \{\, a \in \mathbb{A} \mid \pi(a) \neq a \,\},$$

is a finite set. We shall occasionally make use of the **agreement set**

$$AS(\pi, \pi') \stackrel{\text{def}}{=} \{\, a \in \mathbb{A} \mid \pi(a) = \pi'(a) \,\}$$

and **disagreement set**

$$DS(\pi, \pi') \stackrel{\text{def}}{=} \{\, a \in \mathbb{A} \mid \pi(a) \neq \pi'(a) \,\}.$$

We define the **action of** π **on** E, $\pi \cdot E$, by recursion:

$$\pi \cdot a \stackrel{\text{def}}{=} \pi(a) \quad \pi \cdot \mathsf{P}(E_1, E_2) \stackrel{\text{def}}{=} \mathsf{P}(\pi \cdot E_1, \pi \cdot E_2) \quad \pi \cdot \mathsf{B}([a]E) \stackrel{\text{def}}{=} \mathsf{B}([\pi \cdot a]\pi \cdot E).$$

We define the **action of** π **on** $S \subseteq \mathbb{A}$, denoted by $\pi \cdot S$, to be $\{\, \pi(x) \mid x \in S \,\}$. It is a fact that $(\pi' \circ \pi) \cdot E = \pi' \cdot (\pi \cdot E)$[1] and $id \cdot E = E$ for all permutations and expressions. We will sometimes need to make induction proofs over the size of expressions. We define the **size** $|E|$ of an expression E by recursion:

$$|a| \stackrel{\text{def}}{=} 1 \quad |\mathsf{P}(E_1, E_2)| \stackrel{\text{def}}{=} |E_1| + |E_2| + 1 \quad |\mathsf{B}([a]E)| \stackrel{\text{def}}{=} |E| + 1.$$

If π is a transposition $(a\, a')$ then $\pi \cdot E$ is of course the expression E with *all* occurrences of a and a' *swapped*. The interesting observation [13] (Section 2, page 3) is that α-equivalence can be defined using the notion of atom swapping in lieu of substitution of atoms for occurrences of free variables, i.e. renaming.

A **finite support** of an expression E for the permutation action above is a finite set S of atoms such that if $a \neq a'$ and $a, a' \notin S$ then $(a\, a') \cdot E = E$, for some notion of equality $=$. There is always a smallest such set, $supp(E)$, the (finite) intersection of all finite supports.

We shall not make very much use of this important concept, but since support properties lie at the heart of the key Lemma 6.2, we include the proposition below in order to present two (well known) examples of support. Note that the second part is slightly tangential to the main scope of this paper, since it requires the definition of $\pi \cdot [E]_\alpha$ which we do not make direct use of. The definition is the expected one, well-defined by appeal to Lemma 6.1 and the fact that any permutation is the finite product of transpositions.

Proposition 2.1. 1. *If expressions E are identified up to syntactic equality, then $supp(E)$ is $occ(E)$, the (finite) set of all atoms occurring in E. In particular, if $a, a' \notin occ(E)$, that is $a, a' \not\lhd E$, then $(a\, a') \cdot E \equiv E$.*

2. *If expressions E are identified up to (a notion of) α-equivalence $\sim$, with equivalence classes $[E]_\alpha$, then $supp([E]_\alpha)$ is $free(E)$, the (finite) set of all free atoms in E. In particular, if $a, a' \notin free(E)$, that is $a, a' \# E$, then $(a\, a') \cdot [E]_\alpha = [E]_\alpha$.*

Proof. We shall look at the proof of the proposition for a particular definition $\sim_p$ of α-equivalence, namely Definition 3.1.

The proof of the first part of the proposition is easier than, but analogous to, the proof of the second part which we sketch.

We show that $free(E)$ is a finite support for $[E]_\alpha$ by showing that if $a, a' \# E$ then $(a\, a') \cdot E \sim_p E$. This follow immediately from Lemma 6.2 part 2, since $a, a' \notin free(E)$ implies that $free(E) \subseteq AS((a\, a'), id)$.

[1] We often write $\pi \cdot \pi' \cdot \pi'' \cdot E$ instead of $\pi \cdot (\pi' \cdot (\pi'' \cdot E))$: the action associates to the right.

To show that $supp([E]_\alpha) = free(E)$, suppose there is a support $S \subset free(E)$, in which case $free(E)$ is non-empty. Pick $z \in free(E)$ with $z \notin S$, and any $a \notin free(E)$. Then $a, z \notin S$ and so by assumption $(a\,z) \cdot [E]_\alpha = [E]_\alpha$, implying $(a\,z) \cdot E \sim_p E$. But $free((a\,z) \cdot E) = free(E)$, and since $z \notin free((a\,z) \cdot E)$ because $a \notin free(E)$ then $z \in free(E)$, a contradiction. □

3. Some definitions of α-equivalence

We give five definitions of binary relations on expressions. Each is supposed to capture α-equivalence, and in Section 4 we shall prove the relations pairwise equal. The first two use a *p*ermutation action; the latter two a *r*e-naming axiom.

Definition 3.1. The binary relation $\sim_p$ over the set *Exp* of expressions is inductively defined by

$$\frac{}{a \sim_p a}\ \text{ATOM} \qquad \frac{E_1 \sim_p E_1' \quad E_2 \sim_p E_2'}{\mathsf{P}(E_1, E_2) \sim_p \mathsf{P}(E_1', E_2')}\ \text{PCG}$$

$$\frac{(z\,a) \cdot E \sim_p (z\,b) \cdot E'}{\mathsf{B}([a]E) \sim_p \mathsf{B}([b]E')}\ \text{PI} \quad [z \not\in a, b, E, E'].$$

This definition is analogous to the definition of α-equivalence for λ-expressions in [13] (Section 2, page 3).

Definition 3.2. The binary relation $\sim_{p_\#}$ over the set *Exp* of expressions is inductively defined by

$$\frac{}{a \sim_{p_\#} a}\ \text{ATOM} \qquad \frac{E_1 \sim_{p_\#} E_1' \quad E_2 \sim_{p_\#} E_2'}{\mathsf{P}(E_1, E_2) \sim_{p_\#} \mathsf{P}(E_1', E_2')}\ \text{PCG}$$

$$\frac{(z\,a) \cdot E \sim_{p_\#} (z\,b) \cdot E'}{\mathsf{B}([a]E) \sim_{p_\#} \mathsf{B}([b]E')}\ \text{PI}_\# \quad [z \,\#\, a, b, E, E'].$$

Definition 3.3. The binary relation $\sim_p^1$ over the set *Exp* of expressions is inductively defined by the rules of Definition 3.1 but with the rule PI replaced by

$$\frac{(z\,a) \cdot E \sim_p^1 (z\,b) \cdot E'}{\mathsf{B}([a]E) \sim_p^1 \mathsf{B}([b]E')}\ \text{PI}_1 \quad [z \not\in E, E'].$$

This definition is analogous to the definition of α-equivalence for λ-expressions in [12] (Theorem 2.1, page 216). The notation $\sim_p^1$ arises from three variants $\sim_p^i$ of $\sim_p$ that we consider in Proposition 4.3.

Definition 3.4. The binary relation $\sim_r$ over the set *Exp* of expressions is inductively defined by

$$\frac{}{E \sim_r E}\ \text{REF} \qquad \frac{E_1 \sim_r E_2}{E_2 \sim_r E_1}\ \text{SYM} \qquad \frac{E_1 \sim_r E_2 \quad E_2 \sim_r E_3}{E_1 \sim_r E_3}\ \text{TRS}$$

$$\frac{E_1 \sim_r E_1' \quad E_2 \sim_r E_2'}{\mathsf{P}(E_1, E_2) \sim_r \mathsf{P}(E_1', E_2')}\ \text{PCG} \qquad \frac{E \sim_r E'}{\mathsf{B}([a]E) \sim_r \mathsf{B}([a]E')}\ \text{BCG}$$

$$\frac{}{\mathsf{B}([a]E) \sim_r \mathsf{B}([a']E\{a'/a\})}\ \alpha \quad [a' \not\in a, E].$$

Definition 3.4 is analogous to the definition of α-equivalence for λ-expressions most commonly found in the literature, and certainly in most standard textbooks. One of the first formal presentations is in [2] and the same, though rather less formal approach is taken by [15] (Definition 2.1.11, page 26).

Definition 3.5. The binary relation $\sim_{r_\#}$ over the set *Exp* of expressions is inductively defined by the rules of Definition 3.4 but with the rule α replaced by

$$\frac{}{\mathsf{B}([a]E) \sim_{r_\#} \mathsf{B}([a']E\{a'/a\})}\ \alpha_\# \quad [a' \,\#\, a, E].$$

This is analogous to the definition of α-equivalence for λ-expressions one finds in [16] (Section 1B, page 9).

4. The equality of the definitions

We demonstrate that the five binary relations defined in Section 3 are all the same, each capturing α-equivalence for *Exp*. This section is the heart of the paper: anyone working with these relations, especially anyone undertaking theorem proving, needs to be able to fully understand how and why the definitions yield the same relations. Proofs sketches are available in the literature but are sometimes incorrect, and there seems to be no single source that draws all of the definitions together and properly compares them. That is what we set out to do here.

In fact the very short proof sketch that appears in Proposition 2.2 of [13] is not quite right. Translating the proof into our notation, Gabbay and Pitts say that $\sim_r$ is closed under the rules for $\sim_p$ since $(a\,a')\cdot -$ preserves $\sim_r$. However, this preservation property provides no direct connection between swapping and substitution, which one certainly needs—see our Lemma 6.3. Further, the complete details of the other half of their sketch proof may be conceptually easy (matching our proof at heart), but the formalities are not trivial.

Theorem 4.1. *The relation $\sim_p$ of α-equivalence according to Definition 3.1 is identical to the relation $\sim_{p_\#}$ of Definition 3.2.*

Proof. Let us re-write the rule for $\text{PI}_\#$ as follows, to avoid name clashes with instances of z from PI:

$$\frac{(u\,a)\cdot E \sim_{p_\#} (u\,b)\cdot E'}{\mathsf{B}([a]E) \sim_{p_\#} \mathsf{B}([b]E')}\ \text{PI}_\# \quad [u \# a, b, E, E'].$$

It is trivial that $\sim_p$ is contained in $\sim_{p_\#}$. For the converse we show that $\sim_p$ is closed under the rules for $\sim_{p_\#}$. The only interesting case concerns the rule $\text{PI}_\#$. So suppose that $(u\,a)\cdot E \sim_p (u\,b)\cdot E'$ and $u \# a, b, E, E'$. Now pick any atom $z \neq u$ such that $z \not\in a, b, E, E'$. Using Lemma 6.1 we get

$$(z\,u)\cdot(u\,a)\cdot E \sim_p (z\,u)\cdot(u\,b)\cdot E'.$$

Since non-occurrence obviously implies freshness we have $u, z \# a, b, E, E'$. Hence

$$\begin{array}{l} \mathit{free}(E) \subseteq \mathbb{A} - \{u, z\} = AS((z\,u)\circ(u\,a), (z\,a)) \\ \mathit{free}(E') \subseteq \mathbb{A} - \{u, z\} = AS((z\,u)\circ(u\,b), (z\,b)) \end{array}$$

and from Lemma 6.2 part 2 we get $(z\,u)\cdot(u\,a)\cdot E \sim_p (z\,a)\cdot E$ and $(z\,u)\cdot(u\,b)\cdot E' \sim_p (z\,b)\cdot E'$. From Theorem 4.4 (which appears on page 6) we know that $\sim_p$ is symmetric and transitive, so $(z\,a)\cdot E \sim_p (z\,b)\cdot E'$ and appealing to PI we are done. □

Theorem 4.2. *The relation $\sim_p$ of α-equivalence according to Definition 3.1 is identical to the relation $\sim_p^1$ of Definition 3.3.*

Proof. Let us re-write the rule for PI_1 as follows, to avoid name clashes with instances of z from PI:

$$\frac{(u\,a)\cdot E \sim_p^1 (u\,b)\cdot E'}{\mathsf{B}([a]E) \sim_p^1 \mathsf{B}([b]E')}\ \text{PI}_1 \quad [u \not\in E, E'].$$

It is trivial that $\sim_p$ is contained in $\sim_p^1$. For the converse we show that $\sim_p$ is closed under the rules for $\sim_p^1$. The only interesting case concerns the rule PI_1. If it happens that $u \not\in a, b$ we are done by appeal to PI. Let us without loss of generality assume that $u = a$. So suppose that $(a\,a)\cdot E \sim_p (a\,b)\cdot E'$. We need to show that if $u = a \not\in E, E'$, then $\mathsf{B}([a]E) \sim_p \mathsf{B}([b]E')$. So pick any z with $z \not\in a, b, E, E'$ and we need to show that $(z\,a)\cdot E \sim_p (z\,b)\cdot E'$. Using Lemma 6.1 we have

$$(z\,a)\cdot((a\,a)\cdot E) \sim_p (z\,a)\cdot((a\,b)\cdot E') = ((z\,a)\circ(a\,b))\cdot E'.$$

But since $z, a \not\in E'$ from Lemma 6.2 part 1 we have $((z\,a)\circ(a\,b))\cdot E' = (z\,b)\cdot E'$ and of course $(a\,a)\cdot E = E$ so we are done. □

It is interesting to consider variants of the side condition of the PI rule, and ask which ones also lead to definitions of α-equivalence and which do not. Suppose the relation $\sim_p^i$ is defined by the rules for $\sim_p$ but with the PI rule altered with the following side conditions.

Proposition 4.3. *We consider variants[2] of the relation $\sim_p$ from Definition 3.1 by altering the* PI *rule as follows:*

Rule Name	*Relation Name*	*Condition*	$= \sim_p$
PI	$\sim_p$	$z \not\in a, b, E, E'$	*Y*
PI_1	$\sim_p^1$	$z \not\in E, E'$	*Y*
PI_2	$\sim_p^2$	$z \not\in a, b$	*N*
PI_3	$\sim_p^3$	–	*N*

Then the right hand column indicates when a relation $\sim_p^i$ is equal to $\sim_p$.

[2] The proof of a similar proposition with variants of $\sim_{p_\#}$ is easy and omitted.

Proof. The first two rows of the table follow, of course, from Theorem 4.2.

Rule PI$_2$. Let $a \neq b$ and consider $E \stackrel{\text{def}}{=} \mathsf{P}(a,z)$ and $E' \stackrel{\text{def}}{=} \mathsf{P}(b,a)$ so that

$$\frac{(z\,a)\cdot \mathsf{P}(a,z) = \mathsf{P}(z,a) \sim_p^2 \mathsf{P}(z,a) = (z\,b)\cdot \mathsf{P}(b,a)}{\mathsf{B}([a]\mathsf{P}(a,z)) \sim_p^2 \mathsf{B}([b]\mathsf{P}(b,a))}\ \text{PI}_2 \quad [z \not\in a,b].$$

But clearly these two expressions are not α-equivalent, and more precisely they cannot be related by $\sim_p$ since they can only be derived from PI if $z = a$.

Rule PI$_3$. Let $a \neq b$ and consider $E \stackrel{\text{def}}{=} \mathsf{P}(b,a)$ and $E' \stackrel{\text{def}}{=} \mathsf{P}(a,b)$ so that

$$\frac{(b\,a)\cdot \mathsf{P}(b,a) = \mathsf{P}(a,b) \sim_p^3 \mathsf{P}(a,b) = (b\,b)\cdot \mathsf{P}(a,b)}{\mathsf{B}([a]\mathsf{P}(b,a)) \sim_p^3 \mathsf{B}([b]\mathsf{P}(a,b))}\ \text{PI}_3.$$

But clearly these two expressions are not α-equivalent, and more precisely they cannot be related by $\sim_p$ since they can only be derived from PI if $a = b$. □

Theorem 4.4. *The relation $\sim_p$ of α-equivalence according to Definition 3.1 is identical to the relation $\sim_r$ of Definition 3.4.*

Proof. The proof proceeds by induction.

To show that $\sim_r$ is contained in $\sim_p$ we need to show that $\sim_p$ is closed under the rules for $\sim_r$.

REF We prove by induction over expressions that for any E we have $E \sim_p E$. For atoms a, $a \sim_p a$ follows from ATOM. The induction step for expressions $\mathsf{P}(E_1,E_2)$ is trivial. To show that $\mathsf{B}([a]E) \sim_p \mathsf{B}([a]E)$, we can pick any atom $z \not\in a, E$ and then show that $(z\,a)\cdot E \sim_p (z\,a)\cdot E$. But this is immediate from the induction hypothesis $E \sim_p E$ and Lemma 6.1.

SYM We prove by induction over $\sim_p$ that $(\forall E\ \sim_p\ E')(E'\ \sim_p\ E)$. This is straightforward: In the case of rule PI, if $\mathsf{B}([a]E) \sim_p \mathsf{B}([b]E')$ then there is a particular z such that $(z\,a)\cdot E \sim_p (z\,b)\cdot E'$. By induction we have $(z\,b)\cdot E' \sim_p (z\,a)\cdot E$ and we apply PI again.

BCG (Note: we will require BCG in the proof of TRS below.) Let $E\ \sim_p\ E'$. Pick any atom z where $z \not\in a, E, E'$. From Lemma 6.1 we have $(z\,a)\cdot E \sim_p (z\,a)\cdot E'$ and we are done by PI.

TRS We prove by induction over natural numbers that

$$(\forall n)(\forall E, E', E'')(|E| = |E'| = |E''| = n \Longrightarrow E \sim_p E' \sim_p E'' \Longrightarrow E \sim_p E'').$$

The base step is trivial since for size $n = 1$ only rule ATOM applies and the expressions must actually be identical atoms.

For the induction step, suppose the result holds for all sizes strictly below some size $s > 1$. If $E \sim_p E' \sim_p E''$, by examining the defining rules, we see that the expressions are not atoms, and hence there are outermost constructors that must be the same for each expression.

So suppose that we have $\mathsf{P}(E_1,E_2) \sim_p \mathsf{P}(E_1',E_2') \sim_p \mathsf{P}(E_1'',E_2'')$. Then we must have $E_i \sim_p E_i'$ and $E_i' \sim_p E_i''$ for $i = 1, 2$. By induction we can easily deduce that $E_1 \sim_p E_1''$ and $E_2 \sim_p E_2''$ and we then apply PCG.

Now suppose that $\mathsf{B}([a]E) \sim_p \mathsf{B}([b]E') \sim_p \mathsf{B}([c]E'')$. Hence there must be a $z \not\in a, b, E, E'$ and $z' \not\in b, c, E', E''$ for which

$$(z\,a)\cdot E \sim_p (z\,b)\cdot E' \qquad (z'\,b)\cdot E' \sim_p (z'\,c)\cdot E''.$$

To show that $\mathsf{B}([a]E) \sim_p \mathsf{B}([c]E'')$ let us pick some $u \not\in a, c, E, E'$, and such that $u \not\in b, z, z'$ as well. From Lemma 6.1 we have

$$(u\,z)\cdot(z\,a)\cdot E \sim_p (u\,z)\cdot(z\,b)\cdot E' \quad (u\,z')\cdot(z'\,b)\cdot E' \sim_p (u\,z')\cdot(z'\,c)\cdot E'' \qquad (*)$$

Thus if we can show that

$$(u\,z)\cdot(z\,a)\cdot E \sim_p (u\,a)\cdot E \qquad (\dagger)$$

along with similar relationships for the other three expressions in $(*)$ so that each expression is $\sim_p$-equivalent to one of the form $(u+)\cdot -$, then we can apply the induction hypothesis (*since the size of any expression E equals that of $\rho\cdot E$*) to deduce that $(u\,a)\cdot E \sim_p (u\,c)\cdot E''$ and apply PI. In fact $(\dagger)$ follows easily from Lemma 6.2 part 2, since $u, z \not\in a, E$ implies

$$\mathit{free}(E) \subseteq \mathbb{A} - \{u, z\} = AS((u\,z)\circ(z\,a), (u\,a))$$

with very similar reasoning for the other three relationships analogous to $(\dagger)$. (Note: in fact Lemma 6.2 part 1 applies—each $(\dagger)$ is a syntactic equality!)

PCG Trivial from PCG.

α We must prove that for any atoms a, a' and expressions E, where $a' \not\in a, E$, that we have $\mathsf{B}([a]E) \sim_p \mathsf{B}([a']E\{a'/a\})$. To do this, we need to find an atom $z \not\in a, a', E, E\{a'/a\}$ where $(z\,a)\cdot E \sim_p (z\,a')\cdot E\{a'/a\}$. Thus we prove by induction on E

$$(\forall a, a')(a' \not\in a, E \Longrightarrow (\exists z)(z \not\in a, a', E, E\{a'/a\} \wedge (z\,a)\cdot E \sim_p (z\,a')\cdot E\{a'/a\}).$$

For the base case of an atom b, choose any atom z such that $z \not\in a, a', b$, and suppose further that $a' \not\in a, b$. In the case that $b = a$, we have to prove $z \sim_p z$, and in the case $b \neq a$ we require $b \sim_p b$. Both follow from ATOM.

For the case of an expression $\mathsf{P}(E_1, E_2)$ we need to show that

$$(\forall a, a')(a' \not\in a, \mathsf{P}(E_1, E_2) \Longrightarrow (\exists z)(z \not\in a, a', \mathsf{P}(E_1, E_2), \mathsf{P}(E_1, E_2)\{a'/a\} \wedge (z\ a) \cdot \mathsf{P}(E_1, E_2) \sim_p (z\ a') \cdot \mathsf{P}(E_1, E_2)\{a'/a\}))$$

that is

$$(\forall a, a')(a' \not\in a, \mathsf{P}(E_1, E_2) \Longrightarrow (\exists z)(z \not\in a, a', \mathsf{P}(E_1, E_2), \mathsf{P}(E_1, E_2)\{a'/a\} \wedge \mathsf{P}((z\ a) \cdot E_1, (z\ a) \cdot E_2) \sim_p \mathsf{P}((z\ a') \cdot E_1\{a'/a\}, (z\ a') \cdot E_2\{a'/a\}))).$$

Suppose that $a' \not\in a, \mathsf{P}(E_1, E_2)$, so that $a' \not\in a, E_1, E_2$. By induction there are z_1 and z_2 such that

$$z_i \not\in a, a', E_i, E_i\{a'/a\} \wedge (z_i\ a) \cdot E_i \sim_p (z_i\ a') \cdot E_i\{a'/a\}.$$

We can now choose $u \neq z_i$ and which does not occur in any of the above expressions, and further, appeal to Lemma 6.1 to deduce that there exists u

$$u \not\in a, a', E_1, E_1\{a'/a\}, E_2, E_2\{a'/a\} \wedge (u\ a) \cdot E_i \sim_p (u\ a') \cdot E_i\{a'/a\}$$

since, for example, the non-occurrence conditions mean that by Lemma 6.2 part 2 we have

$$(u\ a) \cdot E_i \sim_p (u\ z_i) \cdot (z_i\ a) \cdot E_i \wedge (u\ a') \cdot E_i\{a'/a\} \sim_p (u\ z_i) \cdot (z_i\ a') \cdot E_i\{a'/a\}.$$

Appealing to PCG we are done.

For the case of a binding expression $\mathsf{B}([b]E)$, choose any atom

$$z \not\in a, a', \mathsf{B}([b]E), (\mathsf{B}([b]E))\{a'/a\}$$

and suppose that $a' \not\in a, \mathsf{B}([b]E)$. We have to show that

$$(z\ a) \cdot \mathsf{B}([b]E) \sim_p (z\ a') \cdot (\mathsf{B}([b]E))\{a'/a\}.$$

(Case $b = a$): Since a is not a free atom in $\mathsf{B}([b]E) \equiv \mathsf{B}([a]E)$, and $z, a' \not\in E$, we have

$$(z\ a') \cdot (\mathsf{B}([a]E))\{a'/a\} \stackrel{\text{def}}{=} (z\ a') \cdot \mathsf{B}([a]E) \stackrel{\text{def}}{=} \mathsf{B}([a]E).$$

Thus we need to show that $\mathsf{B}([z](z\ a) \cdot E) \sim_p \mathsf{B}([a]E)$, and so picking $z' \not\in z, a, (z\ a) \cdot E, E$ it suffices to show that

$$(z'\ z) \cdot ((z\ a) \cdot E) \sim_p (z'\ a) \cdot E.$$

But since $z', z \not\in E$ this follows from Lemma 6.2 part 2; indeed it is an equality by part 1.

(Case $b \neq a$): Since a may be a free atom in $\mathsf{B}([b]E)$, and $a' \not\in \mathsf{B}([b]E)$ implying no renaming on substitution, we have

$$(z\ a') \cdot (\mathsf{B}([b]E))\{a'/a\} \stackrel{\text{def}}{=} (z\ a') \cdot \mathsf{B}([b]E\{a'/a\}).$$

Also $z, a' \neq b$ since $z, a' \not\in \mathsf{B}([b]E)$ so we have to prove that

$$\mathsf{B}([b](z\ a) \cdot E) \sim_p \mathsf{B}([b]((z\ a') \cdot E\{a'/a\}))$$

and this holds provided that by picking $z' \not\in b, (z\ a) \cdot E, (z\ a') \cdot E\{a'/a\}$ we have

$$(z'\ b) \cdot ((z\ a) \cdot E) \sim_p (z'\ b) \cdot ((z\ a') \cdot E\{a'/a\}).$$

By Lemma 6.1 all we need for this is

$$(z\ a) \cdot E \sim_p (z\ a') \cdot E\{a'/a\}$$

which is the induction hypothesis.

To show that $\sim_p$ is contained in $\sim_r$ we need to show that $\sim_r$ is closed under the rules for $\sim_p$.

ATOM Trivial from REF.

PCG Trivial from PCG.

PI Suppose that $(z\ a) \cdot E \sim_r (z\ b) \cdot E'$ where $z \not\in a, b, E, E'$. We appeal to Lemma 6.3 and the assumption together with TRS to obtain $E\{z/a\} \sim_r E'\{z/b\}$. Using BCG we get $\mathsf{B}([z]E\{z/a\}) \sim_r \mathsf{B}([z]E'\{z/b\})$. It follows from α and the condition on z that $\mathsf{B}([a]E) \sim_r \mathsf{B}([z]E\{z/a\})$ and that $\mathsf{B}([b]E') \sim_r \mathsf{B}([z]E'\{z/b\})$. The result follows from SYM applied to the last relationship followed by instances of TRS.

The proof is complete. □

Theorem 4.5. *The relation $\sim_p$ of α-equivalence according to Definition 3.1 is identical to the relation $\sim_{r_\#}$ of Definition 3.5.*

Proof. The format of the proof is (as one would expect) exactly like that of Theorem 4.4. However, some of the technical details require amendment.

To show that $\sim_p$ is contained in $\sim_{r_\#}$ we must show that $\sim_{r_\#}$ is closed under all the rules for $\sim_p$. The steps are exactly like those of the first part of Theorem 4.4. However, instead of Lemma 6.3 we must use Lemma 6.4, and note that instances of α in the proof of closure of the PI rule may be replaced by $\alpha_\#$ since $z \not\in E, E'$ implies $z \# E, E'$.

To show that $\sim_{r_\#}$ is contained in $\sim_p$ we must show that $\sim_p$ is closed under all the rules for $\sim_{r_\#}$. The steps are again exactly like those of the second part of Theorem 4.4, except that the rule α is replaced by $\alpha_\#$. Thus we must prove that for any atoms a, a' and expressions E, where $a' \not\in a, E$, we have $\mathsf{B}([a]E) \sim_p \mathsf{B}([a']E\{a'/a\})$. To do this, we need to find an atom $z \not\in a, a', E, E\{a'/a\}$ where $(z\ a) \cdot E \sim_p (z\ a') \cdot E\{a'/a\}$. In the second part of Theorem 4.4 we proved by induction on E

$$(\forall a, a')(\exists z)(a' \not\in a, E \Longrightarrow (z\ a) \cdot E \sim_p (z\ a') \cdot E\{a'/a\}) \qquad (**)$$

If one examines the proof in Theorem 4.4, the new inductive step for an expression $\mathsf{B}([b]E)$ would have the following case

(Case $b \neq a$): Since a may be a free atom in $\mathsf{B}([b]E)$, and $a' \# \mathsf{B}([b]E)$, there will be a situation where $a' = b$, and hence we have

$$(z\ a') \cdot (\mathsf{B}([b]E))\{a'/a\} \stackrel{\text{def}}{=} (z\ a') \cdot \mathsf{B}([b']E\{b'/b\}\{a'/a\})$$

with a fresh b' substituted for b to avoid capture of new occurrences of a' by $b = a'$. *Thus we will not have a sufficiently strong induction hypothesis:* when the proof is continued, and we get to a point where we try to appeal to an inductive hypothesis $(**)$ involving E, the only hypotheses available will involve a single variable substitution and not multiple ones—so the proof grinds to a halt.

We can get around this problem by proving Lemma 6.5, and then $(**)$ is a simple instance. The proof then proceeds exactly as before. □

Theorem 4.6. *The relation $\sim_r$ of α-equivalence according to Definition 3.4 is identical to the relation $\sim_{r_\#}$ of Definition 3.5.*

Proof. The result follows, of course, from Theorems 4.4 and 4.5. However it is worthwhile considering a direct proof—which, obviously, requires recourse only to traditional renaming techniques.

The proof proceeds by induction. To show that $\sim_r$ is contained in $\sim_{r_\#}$ we must show that $\sim_{r_\#}$ is closed under all the rules for $\sim_r$. This is mainly trivial since most of the rules are the same for each relation. However we do need to show that $\sim_{r_\#}$ is closed under α, that is we need to show that

$$\mathsf{B}([a]E) \sim_{r_\#} \mathsf{B}([a']E\{a'/a\})$$

provided that $a' \not\in a, E$. But this implies the side condition of $\alpha_\#$, namely $a' \# a, E$, and so we are done by using $\alpha_\#$.

To show that $\sim_{r_\#}$ is contained in $\sim_r$ we must show that $\sim_r$ is closed under all the rules for $\sim_{r_\#}$. Again most of the proof is trivial. However the closure of $\sim_r$ under the rule $\alpha_\#$ while conceptually easy and very well understood, is surprisingly tricky from a technical point of view. We have to show that if $a' \# a, E$ then

$$\mathsf{B}([a]E) \sim_r \mathsf{B}([a']E\{a'/a\}).$$

Using Lemma 6.6 we know that there is an expression $\hat{E}$ with $a' \not\in \hat{E}$ and $\hat{E} \sim_r E$. The lemma may be regarded as a form of Barendregt's *variable convention*—we may rename bound variables in E to produce $\hat{E}$ such that there are no occurrences of a in $\hat{E}$. Hence we have

$$\mathsf{B}([a]E) \sim_r \mathsf{B}([a]\hat{E}) \sim_r \mathsf{B}([a']\hat{E}\{a'/a\})$$

using BCG and α. Using Lemma 6.7 we have $\hat{E}\{a'/a\} \sim_r E\{a'/a\}$ and the result follows from BCG and TRS. □

5. Program contexts

The main purpose of this paper is to specify and reason about α-equivalence on expressions. However, in practice, work which involves applications of this theory may well involve, for example, unification techniques, and what are often termed "program contexts".

In [26] (page 480) Urban et al. give a treatment of α-equivalence for λ-expressions with meta-variables (where expressions are defined using free and bound atoms, analogous to our approach). They go on to consider what they call "nominal unification"—a theory of unification founded on expression definitions which make use of a permutation action. In [6] (page 12) one also finds a similar treatment, progressing on to a detailed treatment of re-writing in which matching is defined up to α-equivalence.

Program contexts may be thought of as instances of such expressions in which there is a single distinguished meta-variable. They are fundamental in many definitions of semantic program equivalence (see for example [4,22]), but especially contextual equivalence.

Given the importance of program contexts, we now consider how to define a notion of α-equivalence in the setting of this paper, which slightly simplifies the presentations in [26,6]. In order to do so, and at the same time include new material, we shall

- define a notion of *context* which amounts to the special case of (the notion of) term with meta-variables from [26] (Definition 2.3, page 479) in which there is one distinguished meta-variable, and the notion of position from [6] (page 37);
- write down in Fig. 1 a definition of α-equivalent contexts analogous to that in [26] (Figure 2, page 480) and prove our Theorem 5.1 (compare Lemma 2.14 of [26] (page 484));
- and give a further definition of α-equivalent contexts (in Fig. 2) specified using rules similar to those in Definition 3.1 and show that it is the same as the definition in Fig. 1.

We begin by defining contexts. Since we wish to make definitions involving permutation actions, we will actually consider holes of the form $\pi \cdot -$; we refer to these as **suspensions** in the standard way; for example see [26] (Definition 2.3, page 479).

The set of **contexts** is defined as the subset of the terms generated by the following grammar

$$\mathcal{C} ::= \pi \cdot - \mid a \mid \mathsf{P}(E_1, E_2) \mid \mathsf{B}([a]E)$$

(where a is a meta-variable ranging over $\mathbb{A}$) such that there is at most *one occurrence* of suspension $\pi \cdot -$ in $\mathcal{C}$. There is a permutation action $\pi \cdot \mathcal{C}$ defined recursively by

$$\pi \cdot a \stackrel{\text{def}}{=} \pi(a) \qquad \pi \cdot \rho \cdot - \stackrel{\text{def}}{=} (\pi \circ \rho) \cdot \mathcal{C}$$

$$\pi \cdot \mathsf{P}(\mathcal{C}_1, \mathcal{C}_2) \stackrel{\text{def}}{=} \mathsf{P}(\pi \cdot \mathcal{C}_1, \pi \cdot \mathcal{C}_2) \qquad \pi \cdot \mathsf{B}([a]\mathcal{C}) \stackrel{\text{def}}{=} \mathsf{B}([\pi \cdot a]\pi \cdot \mathcal{C}).$$

If E is an expression, and $\mathcal{C}$ a context, then we write $\mathcal{C}[E]$ for the textual substitution of E for the hole $-$; this may well involve capture of free atoms by any binding atom whose scope includes $-$. The recursive definition of $\pi \cdot \mathcal{C}$ is the obvious one. If $\pi \cdot -$ does occur in $\mathcal{C}$ we shall sometimes write $\mathcal{C}[\pi \cdot -]$. Note that if there is no occurrence of $\pi \cdot -$ in $\mathcal{C}$, then $\mathcal{C}$ is of course a simple expression as defined on page 2. The **size** of a context extends the definition of the size of an expression (see page 3) with $|\pi \cdot X| \stackrel{\text{def}}{=} 1$.

A notion of program context appears in [6] on page 37 where a context is known as a "position". The definition by Fernández and Gabbay seems not quite the same as the one we give, since they exclude non-trivially permuted holes such as $(a\,b) \cdot -$ for $a \neq b$. In [6] the hole occurs once (the same as our definition) but with no actual permutation action—"with trivial moderation" to use their language. One can see that both definitions may be useful in practice, but here we prefer to work with our slightly more general definition.

In order to define a notion of α-equivalence on contexts, we will need to work with assertions about the kinds of atoms that may appear in expressions that are substituted, and assumptions about the hole. To see why, consider for example the equivalence of binding expressions $\mathsf{B}([a]-) \approx \mathsf{B}([b]-)$ $(a \neq b)$, and ask when, provided $E \approx E'$, we have $\mathsf{B}([a]E) \approx \mathsf{B}([b]E')$. Now $u \approx u$ for any atom u; and further $\mathsf{B}([a]u) \approx \mathsf{B}([b]u)$ provided that $a \# u$ and $b \# u$. More generally, one may guess that we require the *assertion* $a \# E$ and $b \# E'$. What about α-equivalence of suspensions, say $(a\,b) \cdot - \approx (c\,d) \cdot -$? Again considering $u \approx u$, we could substitute u for $-$ provided that the action of each permutation is the same on u. Hence u should be different from each of a, b, c, d, that is

$$u \notin \{a, b, c, d\} \qquad (\quad = DS((a\,b), (c\,d)) \quad)$$

recalling that the disagreement set $DS(\pi, \pi') \subset \mathbb{A}$ is the set of atoms on which π and π' differ.

In order to state that a, b, c, d do not occur (fresh) in anything that gets substituted for $-$, we include these atoms as *assumptions* by writing

$$DS((a\,b), (c\,d)) \vdash (a\,b) \cdot - \approx (c\,d) \cdot -.$$

More generally we shall proceed to define two forms of judgment. Let ∇ be a set of atoms. The first judgment form is $\nabla \vdash a \# \mathcal{C}$. In the case that $\mathcal{C}$ contains a simple hole $-$, it is read informally as "on the *assumption* that the atoms in ∇ are fresh for the hole, then we can *assert* a is fresh for $\mathcal{C}$" (see Theorem 5.1). The second is $\nabla \vdash \mathcal{C} \approx \mathcal{C}'$. It is read informally as "on the assumption that the atoms in ∇ are fresh for the hole, then $\mathcal{C}$ and $\mathcal{C}'$ are α-equivalent". The rules defining these judgments appear in Fig. 1 and are close analogues of those in [26]. Note we assume in *the statements of the rules* that $a \neq b$. This is the permutative convention of Gabbay.

It is worth noting that the rule PI_ in Fig. 1 differs from the rule PI in Definition 3.1. Note that the former is defined in terms of a judgment about freshness and the latter non-occurrence. Hence one may ask if we can define α-equivalence on contexts with a rule that mirrors PI. In fact we can, and the rules appear in Fig. 2. Notice that the formalization of freshness assertions has to be replaced by a formalization of non-occurrence.

Urban et al. [26] (Definition 2.3, page 479; and Lemma 2.14, page 484) considered a definition of $\mathcal{C}$ with multiple meta-variables (referred to by them as *terms*). They proved that for any substitution

$$\sigma = [X_1 = E_1, \ldots, X_k = E_k]$$

$$\frac{\pi^{-1}\cdot a \in \nabla}{\nabla \vdash a \# \pi\cdot -}\,{}_{-\#} \qquad \frac{}{\nabla \vdash a \# b} \qquad \frac{\nabla \vdash a \# \mathcal{C}_1 \quad \nabla \vdash a \# \mathcal{C}_2}{\nabla \vdash a \# \mathsf{P}(\mathcal{C}_1, \mathcal{C}_2)}$$

$$\frac{}{\nabla \vdash a \# \mathsf{B}([a]\mathcal{C})} \qquad \frac{\nabla \vdash a \# \mathcal{C}}{\nabla \vdash a \# \mathsf{B}([b]\mathcal{C})}$$

$$\frac{}{\nabla \vdash a \approx a}\,\text{ATOM} \qquad \frac{DS(\pi,\pi') \subseteq \nabla}{\nabla \vdash \pi\cdot - \approx \pi'\cdot -}\,{}_{-}$$

$$\frac{\nabla \vdash \mathcal{C}_1 \approx \mathcal{C}_1' \quad \nabla \vdash \mathcal{C}_2 \approx \mathcal{C}_2'}{\nabla \vdash \mathsf{P}(\mathcal{C}_1,\mathcal{C}_2) \approx \mathsf{P}(\mathcal{C}_1',\mathcal{C}_2')}\,\text{PCG}_{-}$$

$$\frac{\nabla \vdash \mathcal{C} \approx \mathcal{C}'}{\nabla \vdash \mathsf{B}([a]\mathcal{C}) \approx \mathsf{B}([a]\mathcal{C}')}\,\text{BCG}_{-} \qquad \frac{\nabla \vdash \mathcal{C} \approx (a\,b)\cdot \mathcal{C}' \quad \nabla \vdash a \# \mathcal{C}'}{\nabla \vdash \mathsf{B}([a]\mathcal{C}) \approx \mathsf{B}([b]\mathcal{C}')}\,\text{PI}_{-}.$$

In the rules above we take $a \neq b$ as implicit (permutative convention)

Fig. 1. Alpha Equivalent Contexts (#).

$$\frac{\pi^{-1}\cdot a \in \nabla}{\nabla \vdash a \not\sharp \pi\cdot -} \qquad \frac{}{\nabla \vdash a \not\sharp b}$$

$$\frac{\nabla \vdash a \not\sharp \mathcal{C}_1 \quad \nabla \vdash a \not\sharp \mathcal{C}_2}{\nabla \vdash a \not\sharp \mathsf{P}(\mathcal{C}_1,\mathcal{C}_2)} \qquad \frac{\nabla \vdash a \not\sharp \mathcal{C}}{\nabla \vdash a \not\sharp \mathsf{B}([b]\mathcal{C})}$$

$$\frac{}{\nabla \vdash a \sim a} \qquad \frac{DS(\pi,\pi') \subseteq \nabla}{\nabla \vdash \pi\cdot - \sim \pi'\cdot -}$$

$$\frac{\nabla \vdash \mathcal{C}_1 \sim \mathcal{C}_1' \quad \nabla \vdash \mathcal{C}_2 \sim \mathcal{C}_2'}{\nabla \vdash \mathsf{P}(\mathcal{C}_1,\mathcal{C}_2) \sim \mathsf{P}(\mathcal{C}_1',\mathcal{C}_2')} \qquad \frac{\nabla \vdash \mathcal{C} \sim \mathcal{C}'}{\nabla \vdash \mathsf{B}([a]\mathcal{C}) \sim \mathsf{B}([a]\mathcal{C}')}$$

$$\frac{\nabla, z \vdash (z\,a)\cdot \mathcal{C} \sim (z\,b)\cdot \mathcal{C}' \quad \nabla, z \vdash z \not\sharp a, b, \mathcal{C}, \mathcal{C}'}{\nabla \vdash \mathsf{B}([a]\mathcal{C}) \sim \mathsf{B}([b]\mathcal{C}')}\,[z \notin dom(\pi) \cup dom(\pi')]$$

In the rules above we take $a \neq b$ as implicit (permutative convention). Note that $\nabla, z \stackrel{\text{def}}{=} \nabla \cup \{z\}$ and we shall adopt the convention that the notation ∇, z means $z \notin \nabla$. Further $\mathcal{C} \equiv \mathcal{C}[\pi\cdot -]$ and $\mathcal{C}' \equiv \mathcal{C}'[\pi'\cdot -]$

Fig. 2. Alpha Equivalent Contexts ($\not\sharp$).

and equivalence $\mathcal{C} \approx \mathcal{C}'$ one then has $\sigma(\mathcal{C}) \approx \sigma(\mathcal{C}')$. In this paper we work with only one meta-variable (namely $X_1 \stackrel{\text{def}}{=} -$) but do in fact show that if $E_1 \approx E_1'$ then $\mathcal{C}[E_1] \approx \mathcal{C}'[E_1']$ (an analogous result which might have been proven in [26] would require a suitable relation on substitutions, $\sigma \approx \sigma'$, but this is not considered by the authors).

Theorem 5.1. • *Given any* $\nabla \vdash a \# \mathcal{C}$ *and* E *such that for each* $x \in \nabla$ *we have* $\emptyset \vdash x \# E$, *then*

$$\emptyset \vdash a \# \mathcal{C}[E]$$

• *Given any* $\nabla \vdash \mathcal{C} \approx \mathcal{C}'$ *and* $\emptyset \vdash E \approx E'$ *such that for each* $x \in \nabla$ *we have* $\emptyset \vdash x \# E$ *and* $\emptyset \vdash x \# E'$, *then*

$$\emptyset \vdash \mathcal{C}[E] \approx \mathcal{C}'[E'].$$

Proof. The proof proceeds by simultaneous induction on $\nabla \vdash a \# \mathcal{C}$ and $\nabla \vdash \mathcal{C} \approx \mathcal{C}'$.

$\boxed{-_{\#}}$ Suppose that $\nabla \vdash a \# \pi\cdot -$. Then $\pi^{-1}\cdot a \in \nabla$ and hence by assumption $\emptyset \vdash \pi^{-1}\cdot a \# E$. Then by Lemma 6.13 (2) we have $\emptyset \vdash a \# \pi\cdot E$.

$\boxed{-}$ Suppose that $\nabla \vdash \pi\cdot - \approx \pi'\cdot -$ and hence $DS(\pi,\pi') \subseteq \nabla$. We need to prove that $\emptyset \vdash \pi\cdot E \approx \pi'\cdot E'$. From Lemma 6.8 we have $\emptyset \vdash \pi\cdot E \approx \pi\cdot E'$. And from Corollary 6.10 we have $\emptyset \vdash \pi\cdot E' \approx \pi'\cdot E'$. The result follows from Lemma 6.11.

$\boxed{\text{PI}_{-}}$ We need to prove that

$$\emptyset \vdash \mathsf{B}([a]\mathcal{C}[E]) \approx \mathsf{B}([b]\mathcal{C}'[E'])$$

and this follows provided that

1. $\emptyset \vdash \mathcal{C}[E] \approx (a\,b) \cdot \mathcal{C}'[E]$ and
2. $\emptyset \vdash a \# \mathcal{C}'[E']$.

Since $(a\,b) \cdot \mathcal{C}'[E] \equiv ((a\,b) \cdot \mathcal{C}')[E]$ statement 1 follows by induction. And statement 2 follows by induction as well. The inductive steps for the other rules are straightforward. □

Theorem 5.2. *The relation $\nabla \vdash \mathcal{C} \approx \mathcal{C}'$ defined in Fig. 1 is the same as $\nabla \vdash \mathcal{C} \sim \mathcal{C}'$ defined in Fig. 2.*

Proof. Remark: The proof below establishes that the two relations of the theorem are the same. For the purposes of our paper, this is essentially a mathematical point, and we do not consider the complexities of proofs nor make claims about algorithmic advantages. Crucially, though, the relation of Fig. 2 has a rule which includes a name z which is "fresh" for the contexts $\mathcal{C}$, $\mathcal{C}'$ and the set of atoms ∇. We leave as an open question to what extent these rules may have a higher "computational complexity" compared to those of Fig. 1. For a general discussion of unification please see [19].

Recall that ∇, x always means that $x \notin \nabla$. Note that the inductive definitions differ in one rule only. Thus, to show that each relation is closed under the defining rules of the other we need to show that

$$\nabla, z \vdash (z\,a) \cdot \mathcal{C} \approx (z\,b) \cdot \mathcal{C}' \wedge \nabla, z \vdash z \not\in a, b, \mathcal{C}, \mathcal{C}' \wedge z \notin dom(\pi) \cup dom(\pi')$$
$$\Longrightarrow_1 \nabla \vdash \mathcal{C} \approx (a\,b) \cdot \mathcal{C}' \wedge \nabla \vdash a \# \mathcal{C}'$$

and

$$\nabla \vdash \mathcal{C} \sim (a\,b) \cdot \mathcal{C}' \wedge \nabla \vdash a \# \mathcal{C}'$$
$$\Longrightarrow_2 (\exists z)(\nabla \vdash (z\,a) \cdot \mathcal{C} \sim (z\,b) \cdot \mathcal{C}' \wedge \nabla \vdash z \not\in a, b, \mathcal{C}, \mathcal{C}' \wedge z \notin dom(\pi) \cup dom(\pi'))$$

$\boxed{\Longrightarrow_1}$. Suppose that

$$\nabla, z \vdash (z\,a) \cdot \mathcal{C} \approx (z\,b) \cdot \mathcal{C}' \wedge \nabla, z \vdash z \not\in a, b, \mathcal{C}, \mathcal{C}' \wedge z \notin dom(\pi) \cup dom(\pi').$$

First we prove that $\nabla \vdash a \# \mathcal{C}'$.[3] Since $\nabla, z \vdash z \not\in a, b, \mathcal{C}$ we know $\nabla, z \vdash z \# \mathcal{C}$ from Lemma 6.12. Then, since we have $z \neq a$ and $z \neq b$,

$$\begin{array}{lll} \nabla, z \vdash z \# \mathcal{C} \Longrightarrow \nabla, z \vdash a \# (z\,a) \cdot \mathcal{C} & & \text{Lemma 6.13 (2)} \\ \Longrightarrow \nabla, z \vdash a \# (z\,b) \cdot \mathcal{C}' & & \text{Lemma 6.15} \\ \Longrightarrow \nabla, z \vdash a \# \mathcal{C}' & & \text{Lemma 6.13 (1)} \\ \Longrightarrow \nabla \vdash a \# \mathcal{C}' & (*) & \text{Lemma 6.13 (4)} \end{array}$$

where Lemma 6.15 is applied to $\nabla, z \vdash (z\,a) \cdot \mathcal{C} \approx (z\,b) \cdot \mathcal{C}'$.

Establishing the other conjunct $\nabla \vdash \mathcal{C} \approx (a\,b) \cdot \mathcal{C}$ is considerably harder. We prove, by induction on the size of the contexts, that

$$\nabla, z \vdash (z\,a) \cdot \mathcal{C} \approx (z\,b) \cdot \mathcal{C}' \wedge \nabla, z \vdash z \not\in a, b, \mathcal{C}, \mathcal{C}' \wedge z \notin dom(\pi) \cup dom(\pi')$$
$$\Longrightarrow \nabla \vdash \mathcal{C} \approx (a\,b) \cdot \mathcal{C}'.$$

We begin by considering the case when $|\mathcal{C}| = |\mathcal{C}'| = 1$: contexts must be $\pi \cdot -$ or x. In the case of atom x, since $\nabla, z \vdash z \not\in x$ of course $z \neq x$ and the result is trivial by ATOM$_-$. If $x = a$ or $x = b$ the assumption about $\approx$ is false. In the case of a suspension $\pi \cdot -$

$$\nabla, z \vdash (z\,a) \cdot \pi \cdot - \approx (z\,b) \cdot \pi' \cdot - \wedge \nabla, z \vdash z \not\in a, b, \pi \cdot -, \pi' \cdot -.$$

To prove $\nabla \vdash \pi \cdot - \approx (a\,b) \cdot \pi' \cdot -$ we must show that

$$(\forall u)(\pi \cdot u \neq (a\,b) \cdot \pi' \cdot u \Longrightarrow u \in \nabla).$$

For arbitrary u, we have $\pi \cdot u \neq (a\,b) \cdot \pi' \cdot u$ implies

$$(z\,a) \cdot \pi \cdot u \neq (z\,a) \cdot (a\,b) \cdot \pi' \cdot u \tag{1}$$

and hence since $z \notin dom(\pi) \cup dom(\pi')$ we can deduce that $u \neq z$, and so $\pi' \cdot u \neq z$.

[3] Note that a similar argument allows us to deduce $\nabla \vdash b \# \mathcal{C}$. This demystifies the apparent asymmetry arising from the freshness assertion in the rule PI$_-$.

(Case $\pi' \cdot u = a$): Appealing to an instance of $(*)$ we can reason as follows

$$\begin{array}{ll} \nabla \vdash a \# \pi' \cdot - \Longrightarrow \nabla \vdash \pi' \cdot u \# \pi' \cdot - & \\ \Longrightarrow \nabla \vdash u \# - & \text{Lemma 6.13 (3)} \\ \Longrightarrow u \in \nabla. & \end{array}$$

(Case $\pi' \cdot u \neq a$): Since $\pi' \cdot u \neq z$ we have

$$(z\,a) \cdot (a\,b) \cdot \pi' \cdot u = (z\,b) \cdot \pi' \cdot u.$$

Hence from (1) we have $(z\,a) \cdot \pi \cdot u \neq (z\,b) \cdot \pi' \cdot u$, and from the assumption $\nabla, z \vdash (z\,a) \cdot \pi \cdot - \approx (z\,b) \cdot \pi' \cdot -$ and rule $-$ we can deduce $u \in \nabla$.

We must now show that for any $n > 1$, if the result holds for all contexts where $|\mathcal{C}| = |\mathcal{C}'| = r < n$ and $r \geq 1$, then it holds for contexts where $|\mathcal{C}| = |\mathcal{C}'| = n$. We consider details only for binding contexts of the form $\mathsf{B}([v]\mathcal{C})$; the easier details for contexts of the form $\mathsf{P}(\mathcal{C}, \mathcal{C}')$ are omitted. One has to undertake an extensive case analysis on the names of atoms, and we look in detail at one example case.

(Case $x = a$; all other atoms distinct): We must prove that

$$\nabla, z \vdash (z\,a) \cdot \mathsf{B}([a]\mathcal{C}) \approx (z\,b) \cdot \mathsf{B}([y]\mathcal{C}') \wedge \nabla, z \vdash z \not\# a, b, \mathsf{B}([a]\mathcal{C}), \mathsf{B}([y]\mathcal{C}') \wedge \quad z \notin dom(\pi) \cup dom(\pi')$$

$$\Longrightarrow \nabla \vdash \mathsf{B}([a]\mathcal{C}) \approx (a\,b) \cdot \mathsf{B}([y]\mathcal{C}')$$

that is

$$\nabla, z \vdash \mathsf{B}([z](z\,a) \cdot \mathcal{C}) \approx \mathsf{B}([y](z\,b) \cdot \mathcal{C}') \wedge \nabla, z \vdash z \not\# a, b, \mathsf{B}([a]\mathcal{C}), \mathsf{B}([y]\mathcal{C}') \wedge \quad z \notin dom(\pi) \cup dom(\pi')$$

$$\Longrightarrow \nabla \vdash \mathsf{B}([a]\mathcal{C}) \approx \mathsf{B}([y](a\,b) \cdot \mathcal{C}') \qquad (**)$$

From the assumptions we have, together with $\textsc{bcg}_-$, we can deduce that

$$\nabla, z \vdash (z\,a) \cdot \mathcal{C} \approx (z\,y) \cdot (z\,b) \cdot \mathcal{C}' \qquad (***) \qquad \wedge \qquad \nabla, z \vdash z \# (z\,b) \cdot \mathcal{C}'.$$

By Lemma 6.13 (1) we have $\nabla, z \vdash b \# \mathcal{C}'$ and so $\nabla \vdash b \# \mathcal{C}'$ by Lemma 6.13 (4). Since $\nabla, z \vdash z \not\# \mathsf{B}([y]\mathcal{C}')$ we have $\nabla, z \vdash z \not\# \mathcal{C}'$ and hence $\nabla, z \vdash z \# \mathcal{C}'$ by Lemma 6.12. From Lemma 6.9 we see that

$$\nabla, z \vdash b \# \mathcal{C}' \wedge \nabla, z \vdash z \# \mathcal{C}' \Longrightarrow \nabla, z \vdash (z\,b) \cdot \mathcal{C}' \approx \mathcal{C}'$$

and hence from Lemma 6.8, Lemma 6.11 and $(***)$ we have

$$\nabla, z \vdash (z\,a) \cdot \mathcal{C} \approx (z\,y) \cdot \mathcal{C}'.$$

Since in general $|\rho \cdot \mathcal{C}| = |\mathcal{C}|$, we can apply the induction hypothesis to deduce that

$$\nabla \vdash \mathcal{C} \approx (a\,y) \cdot \mathcal{C}'. \tag{2}$$

Using an instance of $(*)$ (from the first part of this theorem) we can deduce that $\nabla \vdash a \# \mathsf{B}([y]\mathcal{C}')$ and so $\nabla \vdash a \# \mathcal{C}'$ since in the current case $a \neq y$. Since also $\nabla \vdash b \# \mathcal{C}'$ we can again apply Lemma 6.9, to deduce $\nabla \vdash (a\,b) \cdot \mathcal{C}' \approx \mathcal{C}'$, and from Lemma 6.8 we have $\nabla \vdash (a\,y) \cdot (a\,b) \cdot \mathcal{C}' \approx (a\,y) \cdot \mathcal{C}'$. This together with (2) and Lemma 6.11 yields

$$\nabla \vdash \mathcal{C} \approx (a\,y) \cdot (a\,b) \cdot \mathcal{C}'.$$

Lemma 6.13 (2) and $\nabla \vdash b \# \mathcal{C}'$ yield $\nabla \vdash a \# (a\,b) \cdot \mathcal{C}'$. Thus by $\textsc{pi}_-$ we have $(**)$ as required

$$\nabla \vdash \mathsf{B}([a]\mathcal{C}) \approx \mathsf{B}([y](a\,b) \cdot \mathcal{C}').$$

$\boxed{\Longrightarrow_2}$. We prove the implication by induction on the size of contexts; this again requires a case analysis on the equality of atoms. We shall consider in detail just one representative case where there are binding contexts and *all atoms are distinct*, showing that

$$\nabla \vdash \mathsf{B}([x]\mathcal{C}) \sim (a\,b) \cdot \mathsf{B}([y]\mathcal{C}') \wedge \nabla \vdash a \# \mathsf{B}([y]\mathcal{C}')$$

$$\Longrightarrow \quad \exists z \left\{ \begin{array}{l} \nabla, z \vdash (z\,a) \cdot \mathsf{B}([x]\mathcal{C}) \sim (z\,b) \cdot \mathsf{B}([y]\mathcal{C}') \\ \nabla, z \vdash z \not\# a, b, \mathsf{B}([x]\mathcal{C}), \mathsf{B}([y]\mathcal{C}') \\ z \notin dom(\pi) \cup dom(\pi') \end{array} \right.$$

that is

$$\nabla \vdash \mathsf{B}([x]\mathcal{C}) \sim \mathsf{B}([y](a\,b) \cdot \mathcal{C}') \wedge \nabla \vdash a \# \mathsf{B}([y]\mathcal{C}')$$

$$\Longrightarrow \quad \exists z \left\{ \begin{array}{l} \nabla, z \vdash \mathsf{B}([x](z\,a) \cdot \mathcal{C}) \sim \mathsf{B}([y](z\,b) \cdot \mathcal{C}') \\ \nabla, z \vdash z \not\# a, b, \mathsf{B}([x]\mathcal{C}), \mathsf{B}([y]\mathcal{C}') \\ z \notin dom(\pi) \cup dom(\pi'). \end{array} \right.$$

So, assuming

$$\nabla \vdash \mathsf{B}([x]\mathcal{C}) \sim \mathsf{B}([y](a\,b)\cdot\mathcal{C}') \wedge \nabla \vdash a \,\#\, \mathsf{B}([y]\mathcal{C}')$$

we have to prove that there are distinct atoms z and z' such that

$$\nabla, z, z' \vdash (z'\,x)\cdot(z\,a)\cdot\mathcal{C} \sim (z'\,y)\cdot(z\,b)\cdot\mathcal{C}' \qquad (\dagger\dagger)$$

$$\nabla, z, z' \vdash z' \not\in x, y, (z\,a)\cdot\mathcal{C}, (z\,b)\cdot\mathcal{C}'$$

$$\nabla, z \vdash z \not\in a, b, \mathsf{B}([x]\mathcal{C}), \mathsf{B}([y]\mathcal{C}')$$

$$z, z' \notin dom(\pi)\cup dom(\pi').$$

From the assumptions we have an atom u such that

$$\nabla, u \vdash (u\,x)\cdot\mathcal{C} \sim (u\,y)\cdot(a\,b)\cdot\mathcal{C}' \wedge \nabla, u \vdash u \not\in x, y, \mathcal{C}, (a\,b)\cdot\mathcal{C}' \wedge \quad u \notin dom(\pi)\cup dom(\pi').$$

We continue with the case[4] when all atoms with different names are distinct, so we have

$$\nabla, u \vdash (u\,x)\cdot\mathcal{C} \sim (a\,b)\cdot(u\,y)\cdot\mathcal{C}' \wedge \nabla, u \vdash u \not\in x, y, \mathcal{C}, (a\,b)\cdot\mathcal{C}'.$$

We can now apply induction (since permutations preserve context size) to deduce that there is a v such that

$$\nabla, u, v \vdash (v\,a)\cdot(u\,x)\cdot\mathcal{C} \sim (v\,b)\cdot(u\,y)\cdot\mathcal{C}'$$

$$\nabla, u, v \vdash v \not\in a, b, (u\,x)\cdot\mathcal{C}, (u\,y)\cdot\mathcal{C}'$$

$$v \notin dom(\pi)\cup dom(\pi').$$

Again, in the case that v is also distinct from all other atoms we have

$$\nabla, u, v \vdash (u\,x)\cdot(v\,a)\cdot\mathcal{C} \sim (u\,y)\cdot(v\,b)\cdot\mathcal{C}'$$

and we take $z' = u$ and $z = v$ so that $(\dagger\dagger)$ is proven. Now all we need to do is verify the remaining non-occurrence conditions. It is immediate that $\nabla, u, v \vdash u \not\in x, y$. Next using Lemma 6.14 (2) and (5) we have

$$\nabla, u \vdash u \not\in \mathcal{C} \Longrightarrow \nabla, u \vdash u \not\in (v\,a)\cdot\mathcal{C} \Longrightarrow \nabla, u, v \vdash u \not\in (v\,a)\cdot\mathcal{C}$$

since $(v\,a)\cdot u = u$. Similarly by Lemma 6.14 (1) and (5) and (2)

$$\nabla, u \vdash u \not\in (a\,b)\cdot\mathcal{C}' \Longrightarrow \nabla, u, v \vdash (a\,b)\cdot u \not\in \mathcal{C}' \Longrightarrow \nabla, u, v \vdash u \not\in (v\,b)\cdot\mathcal{C}'$$

since $(a\,b)\cdot u = u \neq v$ and $u = (v\,b)\cdot u$. It is also immediate that $\nabla, u, v \vdash v \not\in a, b$. And by Lemma 6.14 (1) and $(u\,x)\cdot v = v$ we have

$$\nabla, u, v \vdash v \not\in (u\,x)\cdot\mathcal{C} \Longrightarrow \nabla, u, v \vdash v \not\in \mathcal{C}$$

and further $\nabla, u, v \vdash v \not\in \mathsf{B}([x]\mathcal{C})$ follows from the rules for non-occurrence. Then we use Lemma 6.14 (4) to deduce $\nabla, v \vdash v \not\in \mathsf{B}([x]\mathcal{C})$. Finally, proving $\nabla, v \vdash v \not\in \mathsf{B}([y]\mathcal{C}')$ from $\nabla, u, v \vdash v \not\in (u\,y)\cdot\mathcal{C}$ is similar and omitted. □

6. The lemmas

In this section we present the lemmas used in the paper. For those lemmas mainly involving atom swapping on expressions we give most of the details. For those lemmas whose proofs revolve more centrally around properties of substitution, and hence are found more commonly in the literature, we give only minor comments.

[4] There are of course many cases to consider; the generation of u and indeed v by induction creates of course new cases. Note, though, that $u \neq v$ follows by induction since ∇, u, v follows from the rules.

6.1. Expressions

Lemma 6.1. *For any atoms u and v and expressions E and E' we have*

$$E \sim_p E' \Longrightarrow (u\ v) \cdot E \sim_p (u\ v) \cdot E'.$$

Proof. We use induction over $\sim_p$ for the case $u \neq v$.

ATOM and PCG The details are very easy and omitted.

PI Suppose that $z \not\in a, b, E, E'$. Choose any two atoms $u \neq v$. By induction we have

$$((u\ v) \circ (z\ a)) \cdot E = (u\ v) \cdot ((z\ a) \cdot E) \sim_p (u\ v) \cdot ((z\ b) \cdot E') = ((u\ v) \circ (z\ b)) \cdot E'$$

and we have to prove that

$$(u\ v) \cdot \mathsf{B}([a]E) \sim_p (u\ v) \cdot \mathsf{B}([b]E').$$

There is no choice at this point other than to do a case analysis on the possible equalities between the atoms a, b, u, v, z. Each of a, b and z could be equal to u, or equal to v, or different from both u and v. There are 27 possible such cases. Since z must be different from both a and b, we can dispense immediately with 10 cases: Suppose we take z to be u. Then either $a = u$ with $b = u$, v or $b \neq u, v$, or $b = u$ with $a = u$, v or $a \neq u, v$. There are $3 * 2 - 1 = 5$ *combinations*, and hence $2 * 5$ cases in which z may clash with one of a or b. Here are the other 17 cases:

a	b	z	*Number of cases: Comments*
$\neq$	$\neq$	$\neq$	1
$\neq$	$\neq$	$z = u$	$2 : z = v$ same
$\neq$	$b = u$	$\neq$	$4 : a = u$ and $b \neq u, v$ same; ditto for v
$\neq$	$b = u$	$z = v$	$4 : a = u$ and $b \neq u, v$ and so $z \neq u$ same; and ditto for u, v swapped
$a = u$	$b = u$	$\neq$	2 : ditto for v
$a = u$	$b = u$	$z = v$	$2 : z \neq a, b$; and $a = v$ and $b = v$ same
$a = u$	$b = v$	$\neq$	2 : ditto for u, v swapped

These we now examine.

Each of the four cases where $z \neq u, v$ has a proof similar to the following one:

(Case $a \neq u, v$; $b \neq u, v$; $z \neq u, v$):

We need to prove that

$$\mathsf{B}([a](u\ v) \cdot E) \sim_p \mathsf{B}([b](u\ v) \cdot E').$$

This holds provided there is an atom $\overline{z} \not\in a, b, (u\ v) \cdot E, (u\ v) \cdot E'$ with

$$(\overline{z}\ a) \cdot ((u\ v) \cdot E) \sim_p (\overline{z}\ b) \cdot ((u\ v) \cdot E') \qquad (*)$$

Take $\overline{z} \stackrel{\text{def}}{=} z$ (and in each of the other three cases where $z \neq u, v$ one may take the analogous $\overline{z}$ to be z). Since $z \not\in u, v$, the condition on z implies the desired condition on $\overline{z}$. But $(z\ a) \circ (u\ v) = (u\ v) \circ (z\ a)$ is immediate from the case conditions, and ditto with b for a. The result follows by induction.

Without loss of generality we look at representative examples of the remaining cases:

(Case $a \neq u, v$; $b \neq u, v$; $z = u$):

Again we need to prove $(*)$ above. Take $\overline{z} \stackrel{\text{def}}{=} v$. Since $\overline{z} = v \neq a, b$, $u = z \not\in E, E'$ and $u \neq v$, we have $\overline{z} = v \not\in (u\ v) \cdot E, (u\ v) \cdot E'$. Hence the condition on $\overline{z}$ is satisfied. But $(v\ a) \circ (u\ v) = (u\ v) \circ (v\ a)$ both being (avu), and ditto with b for a. The result follows by induction.

(Case $a \neq u, v$; $b = u$; $z = v$):

We need to prove that

$$\mathsf{B}([a](u\ v) \cdot E) \sim_p \mathsf{B}([v](u\ v) \cdot E').$$

This holds provided there is an atom $\overline{z} \not\in a, v, (u\ v) \cdot E, (u\ v) \cdot E'$ with

$$(\overline{z}\ a) \cdot ((u\ v) \cdot E) \sim_p (\overline{z}\ v) \cdot ((u\ v) \cdot E') \qquad (*)$$

Take $\overline{z} \stackrel{\text{def}}{=} u$. Since $u \neq a$, $u \neq v$, and $v = z \not\in E, E'$ we have $\overline{z} = u \not\in a, v, (u\ v) \cdot E, (u\ v) \cdot E'$. Hence the condition on $\overline{z}$ is satisfied. But $(u\ a) \circ (u\ v) = (u\ v) \circ (v\ a)$ both being (uva), and ditto with b for a. The result follows by induction.

(Case $a = u$; $b = u$; $z = v$):

In this case, the induction hypothesis is $E \sim_p E'$. We need to prove that

$$\mathsf{B}([v](u\ v) \cdot E) \sim_p \mathsf{B}([v](u\ v) \cdot E').$$

This holds provided there is an atom $\overline{z} \not\in v, (u\ v) \cdot E, (u\ v) \cdot E'$ with

$$(\overline{z}\ v) \cdot ((u\ v) \cdot E) \sim_p (\overline{z}\ v) \cdot ((u\ v) \cdot E') \qquad (*)$$

Take $\overline{z} \stackrel{\text{def}}{=} u$. Since $u \neq v$, and $v = z \not\in E, E'$ we have $\overline{z} = u \not\in v, (u\ v) \cdot E, (u\ v) \cdot E'$. Hence the condition on $\overline{z}$ is satisfied, and the result is immediate by induction. □

Lemma 6.2. 1. *For any expression E and permutations π, π',*

$$occ(E) \subseteq AS(\pi, \pi') \qquad \Longrightarrow \qquad \pi \cdot E = \pi' \cdot E.$$

2. *For any expression E and permutations π, π',*

$$free(E) \subseteq AS(\pi, \pi') \qquad \Longrightarrow \qquad \pi \cdot E \sim_p \pi' \cdot E.$$

Proof. The proof of 1. is very similar to 2. and omitted; we prove 2.

First note that for any expression E we can prove that $free(\pi \cdot E) = \pi \cdot free(E)$ by a straightforward induction on E. We omit the details.

We proceed with the lemma by induction on the structure of E. If E is an atom, by assumption the permutations agree on $E = free(E)$ so we appeal to ATOM.

If $E \equiv \mathsf{P}(E_1, E_2)$ the assumption is

$$free(E_1) \cup free(E_2) \subseteq AS(\pi, \pi').$$

Hence $free(E_i) \subseteq AS(\pi, \pi')$ and by induction $\pi \cdot E_i \sim_p \pi' \cdot E_i$. We then appeal to PCG.

If $E \equiv \mathsf{B}([a]E)$ the assumption is

$$free(E) - \{a\} \subseteq AS(\pi, \pi') \qquad (*)$$

We need to show that

$$\pi \cdot \mathsf{B}([a]E) \sim_p \pi' \cdot \mathsf{B}([a]E)$$

that is we need to find an atom z for which $z \# \pi(a), \pi'(a), \pi \cdot E, \pi' \cdot E$ where

$$(z\ \pi(a)) \cdot \pi \cdot E \sim_p (z\ \pi'(a)) \cdot \pi' \cdot E.$$

By induction for E it is enough to show that

$$free(E) \subseteq AS((z\ \pi(a)) \circ \pi, (z\ \pi'(a)) \circ \pi').$$

Pick any $x \in free(E)$. If $x = a$ then x is in the agreement set. If $x \neq a$ then $(*)$ yields that $\pi(x) = \pi'(x)$, and further $\pi(a) \neq \pi(x)$ and $\pi'(a) \neq \pi'(x)$ since permutations are bijections. So it is now enough to show that $z \neq \pi(x)$ (†). The first part of our proof yields $\pi(x) \in \pi \cdot free(E) = free(\pi \cdot E)$ and since $z \notin free(\pi \cdot E)$ we have (†). □

Our next lemma is very similar to Lemma 5.16 of [10]. Indeed, the reader might also wish to consult [9] where, once again, one finds a similar result. In all cases the lemma is key to connecting definitions of α-equivalence based upon permutations with those based upon capture avoiding substitution.

Lemma 6.3. *For any atoms a and z and any expression E*

$$z \not\in E \Longrightarrow (z\ a) \cdot E \sim_r E\{z/a\}.$$

Proof. The result is trivial if $z = a$. For $z \neq a$ we proceed by induction on the structure of E. The steps for atoms and expressions $\mathsf{P}(E_1, E_2)$ are easy.

We consider the induction step for binding expressions $\mathsf{B}([b]E)$. Note that $z \not\in \mathsf{B}([b]E)$ implies that $z \not\in b$ and $z \not\in E$.

(Case $a = b$):

$$\begin{aligned}
(z\ b) \circ \mathsf{B}([b]E) &\stackrel{\text{def}}{=} \mathsf{B}([z](z\ b) \circ E) \\
&\sim_r \mathsf{B}([z]E\{z/b\}) \qquad (\textit{Induction}, \text{BCG}) \\
&\sim_r \mathsf{B}([b]E) \qquad (\alpha) \\
&\stackrel{\text{def}}{=} (\mathsf{B}([b]E))\{z/b\}.
\end{aligned}$$

(Case $a \neq b$):

$$\begin{aligned}
(z\ a) \circ \mathsf{B}([b]E) &\stackrel{\text{def}}{=} \mathsf{B}([b](z\ a) \circ E) \quad (z \neq b) \\
&\sim_r \mathsf{B}([b]E\{z/a\}) \qquad (\textit{Induction}, \text{BCG}) \\
&\stackrel{\text{def}}{=} (\mathsf{B}([b]E))\{z/a\} \quad (z \neq b \quad \textit{so no capture}).
\end{aligned}$$

□

Lemma 6.4. *For any atoms a and z and any expression E*

$$z \not\sharp E \Longrightarrow (z\ a)\cdot E \sim_{r_\#} E\{z/a\}.$$

Proof. As for Lemma 6.3. The only point to note is that the step involving α may be replaced by $\alpha_\#$ since $z \not\sharp E$ implies $z \notin \mathit{free}(E)$. □

Lemma 6.5. *Let I be an index set and let z_i, a_i, a_i' be pairwise disjoint across all $i, j \in I$. Then*

$$(\forall n)(\forall z_i, a_i, a_i')(\forall E)(\quad z_i \not\sharp a_i, a_i', E \wedge a_i' \,\#\, E \Longrightarrow$$

$$(z_1\ a_1)\cdot \ldots \cdot (z_n\ a_n)\cdot E \sim_p (z_1\ a_1')\cdot \ldots \cdot (z_n\ a_n')\cdot E\{a_1'/a_1\}\ldots\{a_n'/a_n\}\quad).$$

Proof. Again the proof is quite long. However, many of the key details depend upon properties of renaming through capture avoiding substitution and are omitted; the nature of the key steps involving swapping can be seen in the proof of Theorem 4.4 on page 6 within the proof steps for rule $\boxed{\alpha}$. □

Lemma 6.6. *For all expressions E and atoms a,*

$$a \,\#\, E \Longrightarrow (\exists \hat{E})(a \not\sharp \hat{E} \wedge \hat{E} \sim_r E).$$

Informally we can always re-name bound atoms so that a particular atom does not occur.

Proof. This result is well known. Note that we induct over the size of expressions. We look briefly at the inductive step for abstractions $\mathsf{B}([b]E)$. If $a \,\#\, \mathsf{B}([b]E)$ then either $a \not\sharp \mathit{free}(E)$ or $a = b$. The former case is easy. For the latter, pick $z \not\sharp a, E$ and then $\mathsf{B}([a]E) \sim_r \mathsf{B}([z]E\{z/a\})$. Since we are inducting over the size of expressions, then we can find E' where $a \not\sharp E'$ and $E' \sim_r E\{z/a\}$, since $a \,\#\, E\{z/a\}$. Then define $\hat{E} \stackrel{\text{def}}{=} \mathsf{B}([z]E')$ and use BCG and TRS. □

Lemma 6.7. *For any expressions E and E′, and any atoms a and b,*

$$E \sim_r E' \Longrightarrow E\{a/b\} \sim_r E'\{a/b\}.$$

Proof. The proof is quite long, but the result is well known and involves only substitution and not swapping. See for example [16]. □

6.2. Contexts

Remark. Most of the lemmas here, that involve $\approx$, appear in [26]. They are included here for completeness. We give some proof details for the results that are new here. The relation $\sim$ is new. The lemmas involving $\sim$ are similar in style to those involving $\approx$, and the proofs similar too.

Lemma 6.8. *For any atoms u and v and contexts $\mathcal{C}$ and $\mathcal{C}'$ we have*

$$\nabla \vdash \mathcal{C} \approx \mathcal{C}' \Longrightarrow \nabla \vdash (u\ v)\cdot \mathcal{C} \approx (u\ v)\cdot \mathcal{C}'$$

$$\nabla \vdash \mathcal{C} \sim \mathcal{C}' \Longrightarrow \nabla \vdash (u\ v)\cdot \mathcal{C} \sim (u\ v)\cdot \mathcal{C}'.$$

Proof. A lengthy rule induction. This is similar to the proof of Lemma 6.1, and for each relation the base case from the rule involving suspensions is straightforward since for any permutation ρ we have $DS(\rho\circ\pi, \rho\circ\pi') = DS(\pi, \pi')$. □

Lemma 6.9. *For any permutations π, π' and freshness assertions $\nabla \vdash x \,\#\, \mathcal{C}$ where $x \in DS(\pi, \pi')$ we have $\nabla \vdash \pi\cdot\mathcal{C} \approx \pi'\cdot\mathcal{C}$. The lemma also applies when $\sim$ replaces $\approx$.*

Proof. An induction over the structure of $\mathcal{C}$. □

Corollary 6.10. *For any permutations π, π' and freshness assertions $\emptyset \vdash x \,\#\, E$ where $x \in DS(\pi, \pi')$ we have $\emptyset \vdash \pi\cdot E \approx \pi'\cdot E$. The lemma also applies when $\sim$ replaces $\approx$.*

Lemma 6.11. *The relation $\nabla \vdash \mathcal{C} \approx \mathcal{C}'$ is transitive. The lemma also applies when $\sim$ replaces $\approx$.*

Proof. This is by induction on the size of the contexts. First one proves that the sizes of α-equivalent contexts are equal. One then shows by induction over the natural numbers that for all n, if $\nabla \vdash \mathcal{C} \approx \mathcal{C}'$ and $\nabla \vdash \mathcal{C}' \approx \mathcal{C}''$ with each context of size n, then $\nabla \vdash \mathcal{C} \approx \mathcal{C}''$. See also [26,6,18] for slightly different approaches to the proof, and a comparison of proof techniques in [25]. □

If one wishes to prove Lemma 6.11 using rule induction, then one must *simultaneously* prove Lemma 6.9. The reason for this is that the inductive hypotheses arising from rule BCG_ will need to be "up to transposition action" and hence one may only proceed by rule induction if, by induction, one may *simultaneously* apply the action of a transposition to an α-equivalence relation.

Lemma 6.12. *For all ∇, a and $\mathcal{C}$ we have $\nabla \vdash a \not\sharp \mathcal{C} \Longrightarrow \nabla \vdash a \,\#\, \mathcal{C}$*

Proof. Since the rules that appear in Fig. 2 are a subset of those in Fig. 1, the statement is immediate. □

Lemma 6.13. *For all ∇, a, z, π, π', $\mathcal{C}$ and $\mathcal{C}'$ we have*

(1) $\nabla \vdash a \# \pi \cdot \mathcal{C} \Longrightarrow \nabla \vdash \pi^{-1} \cdot a \# \mathcal{C}$
(2) $\nabla \vdash \pi \cdot a \# \mathcal{C} \Longrightarrow \nabla \vdash a \# \pi^{-1} \cdot \mathcal{C}$
(3) $\nabla \vdash a \# \mathcal{C} \Longrightarrow \nabla \vdash \pi \cdot a \# \pi \cdot \mathcal{C}$
(4) $\nabla, z \vdash a \# \mathcal{C}[\pi \cdot -] \wedge z \neq a \wedge z \notin dom(\pi) \Longrightarrow \nabla \vdash a \# \mathcal{C}[\pi \cdot -]$

Proof. We can prove the statements (1) and (2) by induction on the structure of $\mathcal{C}$ (together with the fact that permutations are bijections); or we can prove them by an induction over the rules in Fig. 1. Note that (3) is a consequence of (1) and (2). We give details for (4) which does not appear in [26].

$\boxed{\pi \cdot -}$ Suppose that $\nabla, z \vdash a \# \pi \cdot -$. Then $\pi^{-1}(a) \in \nabla \cup \{z\}$. But $\pi(z) = z$ and so since $z \neq a$ we have $z \neq \pi^{-1}(a)$. Hence $\pi^{-1}(a) \in \nabla$ and so $\nabla \vdash a \# \pi \cdot -$.

$\boxed{\mathsf{P}(\mathcal{C}, \mathcal{C}')}$ A trivial induction step.

$\boxed{\mathsf{B}([x]\mathcal{C})}$ The case $a = x$ is immediate from the rules defining freshness; and the case $a \neq x$ follows very simply

$$\nabla, z \vdash a \# \mathsf{B}([x]\mathcal{C}) \Longrightarrow \nabla, z \vdash a \# \mathcal{C} \Longrightarrow \nabla \vdash a \# \mathcal{C} \Longrightarrow \nabla \vdash a \# \mathsf{B}([x]\mathcal{C})$$

with the second implication by induction. □

Lemma 6.14. *For all ∇, a, z, π, π', $\mathcal{C}$ and $\mathcal{C}'$ we have*

(1) $\nabla \vdash a \not\sharp \pi \cdot \mathcal{C} \Longrightarrow \nabla \vdash \pi^{-1} \cdot a \not\sharp \mathcal{C}$
(2) $\nabla \vdash \pi \cdot a \not\sharp \mathcal{C} \Longrightarrow \nabla \vdash a \not\sharp \pi^{-1} \cdot \mathcal{C}$
(3) $\nabla \vdash a \not\sharp \mathcal{C} \Longrightarrow \nabla \vdash \pi \cdot a \not\sharp \pi \cdot \mathcal{C}$
(4) $\nabla, z \vdash a \not\sharp \mathcal{C}[\pi \cdot -] \wedge z \neq a \wedge z \notin dom(\pi) \Longrightarrow \nabla \vdash a \not\sharp \mathcal{C}[\pi \cdot -]$
(5) *For $z \neq a$ we have* $\nabla \vdash a \not\sharp \mathcal{C} \Longrightarrow \nabla, z \vdash a \not\sharp \mathcal{C}$

Proof. The proof is very similar to that of Lemma 6.13. The final statement is trivial to prove by induction over $\mathcal{C}$. The base case for suspensions $\pi \cdot -$ follows, since $\pi^{-1} \cdot a \in \nabla \subset \nabla, z$. □

Lemma 6.15. *For any $\mathcal{C}$, $\mathcal{C}'$, ∇ and a*

$$\nabla \vdash \mathcal{C} \approx \mathcal{C}' \wedge \nabla \vdash a \# \mathcal{C} \Longrightarrow \nabla \vdash a \# \mathcal{C}'.$$

The lemma also applies when $\sim$ replaces $\approx$.

Proof. Proceed by rule induction on the definitions of $\approx$ and $\sim$ making use of Lemmas 6.13 and 6.14. □

7. Conclusions

It has been over a decade since significant advances in the modeling of variable binding were made. As well as the foundational idea of swapping, itself based on the permutation actions of Fraenkel Mostowski set theory [12,13], advances in presheaf models were made by Fiore, Plotkin and Turi [7] and Hofmann [17,11]. During this period, name binding in general has been studied intensively, and a number of definitions of α-equivalence have been proposed. These are spread over the literature, and so it seemed sensible to draw together definitions that illustrate the key use of name swapping, and the renaming axiom, with variant side-conditions. Anyone working in formal semantics in general and theorem proving in particular, will often need to think about the formulations of variable binding that are available, and need to understand how they inter-relate. The hope is that this paper will be especially useful to such people.

It is of course more than reasonable to ask if there are any errors in this paper, especially of the kind that we comment upon in the introduction. It would be foolish, or at least slightly unreasonable, to claim that there are not. As has already been stated, the intention is to provide far more detail than currently available in the literature, but to do so in a mathematically structured way, so the hearts of the proofs are readily accessible. However, the author is currently working on a follow up paper with Luana Fagarasan[5] (a student at Leicester) in which we are coding the proofs presented here. The implementation is being conducted in Isabelle/HOL, and is following the material in this paper as closely as possible, with our proof sketches used as the starting point for each of the mechanized proofs. Berghofer and Urban [1], study different forms of α-equivalence in a theorem prover. Their results are of course related to ours, in that definitions of α-equivalence are compared, but through the *applications* of the definitions. In [27] Vestergaard and Brotherston study the Barendregt convention in a mechanical setting, though their motivation is to mechanize equational proofs about higher-order languages.

Our paper studies (a representative instance of) the notion of α-equivalence. An important question to ask is what an abstract formulation might be, particularly at a level that is wider than, but encompasses, abstract syntax trees. Gabbay studies this question in [8] where he develops a generalization of FM set theory, FMG, and studies how FMG can encompass well-known forms of binding syntax.

[5] We are grateful for the kind support of the Nuffield Foundation through a Research Bursary.

Let $\mathbb{A}$ be the set of atoms. There is a well-defined function

$$Exp \times [\mathbb{A}] \times [\mathbb{A}] \longrightarrow Exp \qquad (E, L', L) \mapsto E\{L'/L\}$$

$$\mathsf{a}_k\{L'/L\} \stackrel{\text{def}}{=} \begin{cases} elt\ (pos\ \mathsf{a}_k\ L)\ L' \text{ if } \mathsf{a}_k \in L \\ \mathsf{a}_k \text{ if } \mathsf{a}_k \notin L \end{cases}$$

$$(\mathsf{P}(E_1, E_2))\{L'/L\} \stackrel{\text{def}}{=} \mathsf{P}((E_1\{L'/L\}), (E_2\{L'/L\}))$$

$$(\mathsf{B}([\mathsf{a}_k]E))\{L'/L\} \stackrel{\text{def}}{=} \begin{cases} \mathsf{B}([\mathsf{a}_k]E\{\overline{L'}/\overline{L}\}) \\ \quad \text{if } (\forall x \in \overline{L}) \begin{pmatrix} x{\downarrow} \vee x \,\#\, E \\ \vee \\ (\mathsf{a}_k \neq elt\ (pos\ x\ \overline{L})\ \overline{L'}) \end{pmatrix} \\ \mathsf{B}([\mathsf{a}_w]E\{\mathsf{a}_w/\mathsf{a}_k\}\{\overline{L'}/\overline{L}\}) \\ \quad \text{if } (\exists x \in \overline{L}) \begin{pmatrix} x{\uparrow} \wedge x \lhd_{\mathrm{fr}} E \\ \wedge \\ \mathsf{a}_k = elt\ (pos\ x\ \overline{L})\ \overline{L'} \end{pmatrix} \end{cases}$$

where

- given lists L' and L of equal length, $\overline{L'}$ and $\overline{L}$ are the same lists in which any occurrences of a_k in L together with their mates in L' are removed; and
- w is chosen to be the maximum of the indices occurring E, L' and L, plus 1; note that as $\mathsf{a}_k = elt\ (pos\ x\ \overline{L})\ \overline{L'}$ holds in the clause involving w, then $w > k$.

Fig. A.3. Simultaneous atom substitution

I would like to finish by thanking the anonymous referees whose comments have led to significant improvements in this paper, especially the treatment of program contexts, and the overall structure and exposition.

Appendix A. Simultaneous capture avoiding atom substitution

We assume that our set of atoms $\mathbb{A}$ is linearly ordered by the vertical order on the natural numbers. Thus each atom may be understood to be of the form a_k where a here is a fixed symbol and k is a natural number. Let L and L' be lists of equal length, where L and L' are lists of atoms. Suppose that $\mathsf{a}_{k'} \in L'$ and $\mathsf{a}_k \in L$ both occur at some position p. Then we shall call the expression and variable **mates**, and say that one is the **mate** of the other. If $\mathsf{a}_k \in L$, then we refer to the first occurrence as **active**, written $\mathsf{a}_k{\uparrow}$. Any other occurrences are referred to as **inactive**, written $\mathsf{a}_k{\downarrow}$. We write $E\{L'/L\}$ for, informally, the simultaneous capture avoiding substitution of each atom $\mathsf{a}_{k'} \in L'$ for free occurrences in E of its mate in L. The definition of the simpler $E\{\mathsf{a}_{k'}/\mathsf{a}_k\}$ is then immediate. Note that some free variables in E may have multiple occurrences in L; if so, the expression in L' which is the mate of the active occurrence is the one that is substituted—see the formal definition below.

We need to define atom substitution to be a function on syntax. We have to take great care with the definition of capture avoiding substitution on abstractions where a renaming takes place, in particular with the choice of the renaming variable. The definition is in Fig. A.3. Note that $elt\ n\ L$ is the atom at position n in L if this exists, and that $pos\ \mathsf{a}_k\ L$ is the position of a_k in L if this exists.

Theorem A.1. *The capture avoiding substitution function with action*

$$(E, L', L) \mapsto E\{L'/L\}$$

is specified by recursion over the inductively defined set Exp in Fig. A.3. This function exists.

Proof. The existence of the function follows from standard properties of *Exp* and the defining equations. See for example [5] (Appendix A, page 623). □

References

[1] S. Berghofer, C. Urban, A head-to-head comparison of de Bruijn indices and names, Electronic Notes In Theoretical Computer Science 174 (5) (2007) 53–67.
[2] A. Church, The Calculi of Lambda Conversion, Princeton University Press, 1941.
[3] R. Clouston, Equational logic for names and binders, Ph.D. Thesis, University of Cambridge Computer Laboratory, 2010.

[4] R.L. Crole, Completeness of bisimilarity for contextual equivalence in linear theories, Electronic Journal of the IGPL 9 (1) (2001).
[5] R.L. Crole, Representational adequacy for hybrid, Mathematical Structures in Computer Science 21 (3) (2011) 585–646.
[6] M. Fernández, M. Gabbay, Nominal rewriting, Information and Computation 205 (6) (2007) 917–965.
[7] M. Fiore, G.D. Plotkin, D. Turi, Abstract syntax and variable binding, in: Proceedings of the 14th Annual Symposium on Logic in Computer Science, LICS'99, IEEE Computer Society Press, 1999, pp. 193–202.
[8] M. Gabbay, A general mathematics of names, Information and Computation 205 (7) (2007) 982–1011.
[9] M. Gabbay, A. Mathijssen, Capture-avoiding substitution as a nominal algebra, Formal Aspects of Computing 20 (4-5) (2008) 451–479.
[10] M.J. Gabbay, Foundations of nominal techniques: logic and semantics of variables in abstract syntax, Bulletin of Symbolic Logic 17 (2) (2011) 161–229.
[11] M.J. Gabbay, M. Hofmann, Nominal renaming sets, in: Logic for Programming, Artifical Intelligence, and Reasoning, 2008, pp. 158–173.
[12] M.J. Gabbay, A.M. Pitts, A new approach to abstract syntax involving binders, in: G. Longo (Ed.), Proceedings of the 14th Annual Symposium on Logic in Computer Science, LICS'99, IEEE Computer Society Press, Trento, Italy, 1999, pp. 214–224.
[13] M.J. Gabbay, A.M. Pitts, A new approach to abstract syntax with variable binding, Formal Aspects of Computing 13 (2002) 341–363.
[14] H.B. Curry, R. Feys, Combinatory Logic, vol. I., North-Holland, 1958.
[15] H. Barendregt, The Lambda Calculus: Its Syntax and Semantics, in: Studies in Logic and the Foundations of Mathematics, vol. 103, North Holland, 1984.
[16] J. Hindley, J. Seldin, Introduction to Combinators and the Lambda Calculus, in: London Mathematical Society Student Texts, vol. 1, Cambridge University Press, 1988.
[17] M. Hofmann, Semantical analysis of higher-order abstract syntax, in: Proc. of 14th Ann. IEEE Symp. on Logic in Computer Science, LICS'99, Trento, Italy, 2–5 July 1999, IEEE Computer Society Press, Los Alamitos, CA, 1999, pp. 204–213.
[18] R. Kumar, M. Norrish, (Nominal) Unification by recursive descent with triangular substitutions, in: M. Kaufmann, L.C. Paulson (Eds.), Interactive Theorem Proving, First International Conference, ITP 2010, in: Lecture Notes in Computer Science, vol. 6172, Springer, 2010, pp. 51–66.
[19] J. Levy, M. Villaret, Nominal unification from a higher-order perspective, Computing Research Repository (2010) abs/1005.3731.
[20] J. McKinna, R. Pollack, Some lambda calculus and type theory formalized, Journal of Automated Reasoning 23 (1999) 373–409.
[21] M. Norrish, R. Vestergaard, Proof pearl: de Bruijn terms really do work, in: K. Schneider, J. Brandt (Eds.), Theorem Proving in Higher Order Logics, 20th International Conference, in: Lecture Notes in Computer Science, vol. 4732, springer, 2007, pp. 207–222.
[22] A.M. Pitts, Operationally-based theories of program equivalence, in: P. Dybjer, A.M. Pitts (Eds.), Semantics and Logics of Computation, Cambridge University Press, 1995, Based on lectures given at the CLICS-II Summer School on Semantics and Logics of Computation, Isaac Newton Institute for Mathematical Sciences, Cambridge UK, September 1995.
[23] A.M. Pitts, Nominal logic, a first order theory of names and binding, Information and Computation 186 (2003) 165–193.
[24] R. Pollack, The theory of LEGO: a proof checker for the extended calculus of constructions, Ph.D. Thesis, University of Edinburgh, 1994.
[25] C. Urban, Nominal unification revisited, in: Proceedings 24th International Workshop on Unification, 2010, pp. 1–11.
[26] C. Urban, A.M. Pitts, M.J. Gabbay, Nominal unification, Theoretical Computer Science 323 (2004) 473–497.
[27] R. Vestergaard, J. Brotherston, The mechanisation of barendregt-style equational proofs (the residual perspective), Electronic Notes In Theoretical Computer Science 58 (1) (2001) 18–36.