Two-and-a-halfth order lambda-calculus

Murdoch J. Gabbay

http://www.gabbay.org.uk

Dominic P. Mulligan

http://www.macs.hw.ac.uk/~dpm8

Abstract

Two-and-a-halfth order lambda-calculus is designed to provide a mathematical model of capturing substitution, which is a feature of the informal meta-level. There are two levels of variable; atoms representing object-level variables, and unknowns representing meta-variables. Lambda-abstraction and beta-reduction exist for both atoms and unknowns. This extends the two-levels of variable in nominal terms with a functional meaning.

Keywords: Lambda Calculus, Meta-variables, Functional programming, Confluence, Nominal terms

1 Introduction

The λ-calculus is a syntax to express function abstraction and application which is simple, amenable to mathematical analysis, and easy to extend. However, not everything that looks like a function fits obviously into the λ-calculus; examples include *capturing substitution* and functions depending on intensional properties like free variables and freshness conditions associated with free variables. This paper defines a λ-calculus which includes an explicit model of these notions.

Capturing substitution and freshness are quite common, especially at the 'informal meta-level' — the prose discourse of, for example, this paper. Here are some examples:

- λ-calculus: $(\lambda x.r)[y \mapsto t] = \lambda x.(r[y \mapsto t])$ if x is fresh for t
- π-calculus: $\nu x.(P \mid Q) = P \mid \nu x.Q$ if x is fresh for P
- First-order logic: $\forall x.(\phi \supset \psi) = \phi \supset \forall x.\psi$ if x is fresh for ϕ

These quoted (reified) informal statements mention two levels of variable; *object-variables* x, y and *meta-variables* r, t, P, Q, ϕ, ψ. Capture-avoidance conditions are (freshness) constraints on the values that meta-variables may assume.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

It seems that meta-variables are naturally substituted with capturing substitution; consider the following quote:

"Set r to x and t to x in $(\lambda x.r)[y \mapsto t]$; obtain $(\lambda x.x)[y \mapsto x]$."

Capturing substitution on syntax trees is not hard to define, yet syntax usually has semantics, which motivates the study of (amongst other things) α-equivalence, unification, and β-equivalence.

This work extends nominal terms [20]. Nominal terms feature a two-level hierarchy of variables reflecting the hierarchy noted above: **level 1** variables $a, b, c, d, \ldots$ (atoms) model object-variables; **level 2** variables $X, Y, Z, \ldots$ (unknowns) model meta-variables. In the terminology of this paper, nominal terms' contribution was to understand α-equivalence of level 1 variables in the presence of level 2 variables (object-level α-equivalence in the presence of meta-variables).

Nominal terms can be the basis of expressive, mathematically well-behaved, and implementable systems. They were introduced in a unification algorithm (freshness conditions are constraints) [20], then extended to rewriting [5], an implemented logic programming language [3], and used as a basis of universal algebra [16].

Variables in nominal terms have no in-built functional meaning. There is no λ-abstraction λX or λa, and no application. In a suitable (generalised) sense, nominal terms are first-order. In this paper we extend nominal terms to investigate full $\alpha\beta$-equivalence in the presence of level 1 and level 2 variables ($\alpha\beta$-equivalence in the presence of object- and meta-variables).

We present **two-and-a-halfth order λ-calculus**. We give a syntax, reduction system, and prove confluence and consistency. The calculus contains the untyped λ-calculus (two copies, in fact; at levels 1 and level 2), so it is not strongly normalising.

Just as the step from first-order terms to higher-order terms implies an increase in expressivity traded off against the loss of some computational and mathematical properties [22], so we expect similar benefits and drawbacks moving from nominal terms to two-and-a-halfth order λ-calculus.

Also, until now the only semantics for level 2 variables (unknowns) in nominal terms was 'they range over terms'. This purely syntactic meaning is acceptable for unification, matching, and rewriting [20,5], and possibly for logic-programming [3], but taking a broader view this is clearly only a partial answer. By developing notions of λ-abstraction and β-reduction for level 2 variables we take a step towards understanding the deeper meaning of nominal techniques.

We sketch with examples the kind of thing we can do, or believe should be possible in future work, using two-and-a-halfth order λ-calculus. Syntax and full definitions follow.

"Set t to be x in $\lambda x.t$" and "set t to be y in $\lambda x.t$" are modelled by reductions

$$(\lambda X.(\lambda a.X))a \to (\lambda a.X)[X := a] \equiv \lambda a.a \quad \text{and}$$
$$(\lambda X.(\lambda a.X))b \to (\lambda a.X)[X := b] \equiv \lambda a.b.$$

($\equiv$ is syntactic equivalence, i.e. 'the same term'.) Capture-avoidance for λa (level 1) in the presence of level 2 variables is managed using permutations and freshness in nominal terms style: $c\#X \vdash (\lambda b.(\lambda a.X))a \to \lambda c.(((c\ a) \cdot X)[b \mapsto a])$.

"Set r to x and t to x in $(\lambda x.r)[y{\mapsto}t]$; obtain $(\lambda x.x)[y{\mapsto}x]$" is expressed by:

"$\lambda Z.\lambda Y.((\lambda a.Z)[b \mapsto Y])$ applied to a twice reduces to $(\lambda a.a)[b \mapsto a]$."

It is not hard to impose a simple type system on our syntax (this should be a special case of [4]; here, λX should cause no essential new difficulties). An interesting application for our calculus is as the basis of a logic. Example axioms are (we indicate types with subscripts):

- $\forall P_o.(a_o \# P_o \supset P_o \supset \forall a_o.P_o)$
  Here o is a type of truth-values. $\forall$ is short for $\forall\lambda$ where $\forall$ is a constant symbol. $\#$ is short for $\#\lambda$ where $\#$ is a constant symbol intended to internalise the nominal freshness judgement. This models 'for all ϕ, if $a \notin fv(\phi)$ then $\phi \supset \forall a.\phi$'.
- $\forall X_\alpha.(a_\beta \# X_\alpha \supset \lambda a_\beta.(X_\alpha a_\beta) = X_\alpha)$
  Here $=$ is a constant symbol, written infix. α and β are intended to be arbitrary types. This models η-equivalence (extensionality) at level 1.
- $\forall P_o.(И a_{\mathbb{A}}.\neg P_o) \Leftrightarrow \neg И a_{\mathbb{A}}.P_o$.
  Here И is short for Иλ where И is a constant symbol intended to internalise the Gabbay-Pitts 'new' quantifier [13]. $\neg$ and $\Leftrightarrow$ are constant symbols. $\mathbb{A}$ is a 'type of atoms' with no term-formers. This models the self-duality of И.

The reductions are real reductions in our calculus. The axioms are expressed in a logic which does not yet exist, but they have some mathematical force because the structure they express has been studied in previous work with level 2 variables but (since nominal terms have no λX) without a level 2 quantification explicitly represented in the syntax. The literature contains axiom-sets for several specific object languages, including first-order logic, the π-calculus, and subsets of ML [16,13,5,8].

2 The syntax

Fix disjoint countably infinite sets of **level 1** and **level 2** variables. We let $a, b, c, \ldots$ and $X, Y, Z, \ldots$ range over level 1 and level 2 variables respectively. We use a **permutative convention** that these are distinct. For example, 'a and b' means any pair of distinct level 1 variables.

A **permutation** π is a *finitely supported* bijection of level 1 variables. Here 'finitely supported' means that $\pi(a) = a$ for all but finitely many level 1 variables. Write id for the identity permutation, so $\mathrm{id}(a) = a$ always; write $\circ$ for functional composition, so $(\pi \circ \pi')(a) = \pi(\pi'(a))$; write π^{-1} for inverse; and write $(a\ b)$ for the **swapping** such that $(a\ b)(a) = b$, $(a\ b)(b) = a$, and $(a\ b)(c) = c$.

Definition 1 Let the syntax of two-and-a-halfth-order λ-calculus be:

$$r, s, t, u, v \quad ::= \quad a \mid \pi \cdot X \mid \lambda a.r \mid \lambda X.r \mid rr$$

We equate terms up to α-equivalence of λX-bound variables, but *not* λa-bound variables. We write $\equiv$ for syntactic equivalence (up to α-equivalence of λX-bound variables). For example, $\lambda X.X \equiv \lambda Y.Y$ but $\lambda a.X \not\equiv \lambda b.X$ and $\lambda a.a \not\equiv \lambda b.b$. There is no paradox here. We do not seek to eliminate the meta-level; we want mathematical models with which to study it. That is why we have given our calculus two levels

of variable.

We may write $r[a \mapsto t]$ as shorthand for $(\lambda a.r)t$.

Define a notion $fv(r)$ of **free occurrence** (for level 2) by:

$$fv(a) = \{\} \quad fv(\pi \cdot X) = \{X\} \quad fv(r'r) = fv(r') \cup fv(r)$$
$$fv(\lambda a.r) = fv(r) \quad fv(\lambda X.r) = fv(r) \setminus \{X\}$$

Definition 2 Define **level 2 substitution** $r[X := t]$ and **level 1 permutation** actions by the rules in Figure 1.

Level 2 substitution avoids capture by λ-abstracted level 2 variables, but not by λ-abstracted level 1 variables; compare the clauses for $(\lambda a.r)[X := t]$ and $(\lambda Y.r)[X := t]$.

$\pi \cdot X$ is a *moderated* level 2 variable. Moderations 'rename' level 1 variables in uninstantiated level 2 variables; for example:

$$((b\ a) \cdot X)[X{:=}a] \equiv b$$

This is the treatment of α-equivalence from nominal terms; for a discussion of advantages of this approach see Cheney's comment that permutations are 'inherently capture-avoiding' [2] and the surrounding discussion.

Definition 2 must decide how permutation π interacts with λX. The intuition is 'π moderates free but not bound level 2 variables'. For example:

$$\pi \cdot (XY) \equiv (\pi \cdot X)(\pi \cdot Y) \qquad \pi \cdot \lambda X.(XY) \equiv \lambda X.X\pi \cdot Y$$
$$\pi \cdot \lambda Y.\lambda X.(XY) \equiv \lambda Y.\lambda X.(XY).$$

$Y \notin fv(t)$ in the clause for $(\lambda Y.r)[X := t]$ can be guaranteed by renaming Y.

The definitions of $r[X := t]$ and $\pi \cdot r$ are intertwined. This is due to the clause for $\pi \cdot (\lambda X.r)$. The proof that $\pi \cdot r$ and $r[X := t]$ are well-defined is routine and we omit it. It uses a notion of **depth** of a term, which will be useful later:

$$depth(a) = 1 \qquad depth(\pi \cdot X) = 1 \qquad depth(\lambda a.r) = 1 + depth(r)$$
$$depth(\lambda X.r) = 1 + depth(r) \qquad depth(r'r) = depth(r') + depth(r)$$

Lemma 3 $r[X := \pi \cdot X][X := \pi' \cdot X] \equiv r[X := (\pi \circ \pi') \cdot X]$

Lemma 4 $\pi \cdot (\pi' \cdot r) \equiv (\pi \circ \pi') \cdot r$

Lemma 5 $\pi \cdot (r\sigma) \equiv (\pi \cdot r)\sigma$

Proof We prove the result for $\sigma = [Y := u]$ by induction on r. We consider just the case of $\lambda X.r$: α-converting X if necessary, assume $X \notin fv(u)$. Then:

$$(\pi \cdot \lambda X.r)[Y := u] \equiv \lambda X.(\pi \cdot r[X := \pi^{-1} \cdot X][Y := u])$$
$$\pi \cdot ((\lambda X.r)[Y := u]) \equiv \pi \cdot (\lambda X.r[Y := u]) \equiv \lambda X.(\pi \cdot r[X := \pi^{-1} \cdot X][Y := u]).$$

□

Lemma 6 $depth(r) = depth(\pi \cdot r)$.

3 Freshness and reductions

Definition 7 A **freshness** is a pair $a\#r$. Call $a\#X$ a **primitive freshness**. Call a finite set of primitive freshnesses Δ a **freshness context**. We may drop set brackets, for example writing $\{a\#X, b\#X\}$ as $a\#X, b\#X$ and writing $\Delta, a\#X$ for $\Delta \cup \{a\#X\}$. We may also write $a\#X, b\#X$ as $a, b\#X$. Figure 2 gives freshnesses a notion of derivation.

The notion of nominal freshness is well-discussed in the literature [13,20,16]. Intuitively, $a\#r$ corresponds with '$a \notin fv(r)$', read 'a is abstracted/not free in r'. In the presence of unknowns, which can be instantiated (substituted in a possibly capturing manner) to any term, the notion of 'free variables' is replaced by a notion of freshness which may depend on freshness assumptions on unknowns. For example:

$$\dfrac{\dfrac{\dfrac{}{a\#X, a\#Y \vdash a\#X}\,(\mathbf{a\#X}) \quad \dfrac{}{a\#X, a\#Y \vdash a\#Y}\,(\mathbf{a\#X})}{a\#X, a\#Y \vdash a\#XY}\,(\mathbf{a\#app})}{a\#Y \vdash b\#\lambda X.(X(b\ a)\cdot Y)}\,(\mathbf{a\#\lambda X}) \qquad \dfrac{\dfrac{}{a\#X \vdash a\#\lambda a.X}\,(\mathbf{a\#\lambda a})}{\vdash a\#\lambda X.\lambda a.X}\,(\mathbf{a\#\lambda X})$$

We may write '$\Delta \vdash a\#t$' as shorthand for '$\Delta \vdash a\#t$ is derivable'. We use this shorthand for other derivable assertions including $\rightarrow$, $\rightarrow^*$, $\Rightarrow$, $\Rightarrow^*$, defined later.

$(\mathbf{a\#\lambda X})$ does not implement the denotational notion of freshness for functions from nominal sets [13], with intuition "$(b\ a)\cdot\lambda X.r$ equals $\lambda X.r$ for fresh b". Informally, a is fresh for the function '$\lambda X.\lambda a.X$' in the sense of $(\mathbf{a\#\lambda X})$, but not fresh in the sense of nominal sets. $(\mathbf{a\#\lambda X})$ is a strictly weaker condition; a discussion of denotations is for a later paper.

$(\mathbf{a\#\lambda X})$ is syntax directed, in the sense that $depth(\pi\cdot\lambda X.r) = depth(r)+1 > depth(r) = depth(\pi\cdot r)$ (see Lemma 6).

Definition 8 Write $\Delta \vdash \text{-} \triangleright \text{-}$ for a binary relation on terms parameterised on Δ. Call $\Delta \vdash \text{-} \triangleright \text{-}$ a **congruence** when it is closed under the rules in Figure 3.

Remark 9 $(\triangleright\alpha)$ builds in the nominal term notion of α-equivalence for level 1 variables (see [10, Lemma 3.2]). A better name for congruence might be 'congruent α-equivalence', but we just write 'congruence'.

Definition 10 Write $level(t) = 2$ if t mentions a level 2 variable, free or bound. Otherwise, write $level(t) = 1$. For example $level((a\ b)\cdot X) = level(\lambda X.X) = 2$ and $level(a) = level(\lambda a.a) = 1$.

Write $a \notin \Delta$ when $a\#X \notin \Delta$ for all level 2 variables X. Write $X \notin \Delta$ when $a\#X \notin \Delta$ for all level 1 variables a.

We can now define our reduction relation:

Definition 11 Let $\Delta \vdash \text{-} \rightarrow \text{-}$ be the least congruence closed under the rules in Figure 4.

Recall that $r[a \mapsto t]$ is sugar for $(\lambda a.r)t$. Note that level 2 β-reduction is a single step $[X := t]$, level 1 β-reduction is not. Definition 22 gives the single step level 1 β-reduction. There are no level 3 variables in this particular calculus and therefore no special need arises to break level 2 β-reduction into smaller steps.

Example reductions are in the Introduction. Note that capture-avoiding substi-

$$a[X := t] \equiv a \qquad (\pi \cdot X)[X := t] \equiv \pi \cdot t \qquad (\pi \cdot Y)[X := t] \equiv \pi \cdot Y$$
$$(\lambda a.r)[X := t] \equiv \lambda a.(r[X := t]) \qquad (r'r)[X := t] \equiv (r'[X := t])(r[X := t])$$
$$(\lambda Y.r)[X := t] \equiv \lambda Y.(r[X := t]) \quad (Y \notin fv(t))$$

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \qquad \pi \cdot (r'r) \equiv (\pi \cdot r')(\pi \cdot r)$$
$$\pi \cdot (\lambda a.r) \equiv \lambda \pi(a).(\pi \cdot r) \qquad \pi \cdot (\lambda X.r) \equiv (\lambda X.\pi \cdot r[X := \pi^{-1} \cdot X])$$

Figure 1. Level 2 substitution and level 1 permutation action (Definition 2)

$$\frac{}{\Delta \vdash a\#b}\ (\mathbf{a\#b}) \qquad \frac{}{\Delta \vdash a\#\lambda a.r}\ (\mathbf{a\#\lambda a}) \qquad \frac{\Delta \vdash a\#r}{\Delta \vdash a\#\lambda b.r}\ (\mathbf{a\#\lambda b}) \qquad \frac{\pi^{-1}(a)\#X \in \Delta}{\Delta \vdash a\#\pi\cdot X}\ (\mathbf{a\#X})$$

$$\frac{\Delta \vdash a\#r' \quad \Delta \vdash a\#r}{\Delta \vdash a\#r'r}\ (\mathbf{a\#app}) \qquad \frac{\Delta, a\#X \vdash \pi(a)\#\pi\cdot r \quad (X \notin \Delta)}{\Delta \vdash \pi(a)\#\pi\cdot(\lambda X.r)}\ (\mathbf{a\#\lambda X})$$

Figure 2. Freshness entailment (Definition 7)

$$\frac{\Delta \vdash r \triangleright s}{\Delta \vdash \lambda a.r \triangleright \lambda a.s}\ (\triangleright\lambda\mathbf{a}) \qquad \frac{\Delta \vdash r \triangleright s \quad \Delta \vdash t \triangleright u}{\Delta \vdash rt \triangleright su}\ (\triangleright\mathbf{app})$$

$$\frac{\Delta \vdash r \triangleright s \quad (X \notin \Delta)}{\Delta \vdash \lambda X.r \triangleright \lambda X.s}\ (\triangleright\lambda\mathbf{X}) \qquad \frac{\Delta \vdash r \triangleright s \quad \Delta \vdash a\#s \quad \Delta \vdash b\#s}{\Delta \vdash r \triangleright (a\ b)\cdot s}\ (\triangleright\alpha)$$

Figure 3. Congruence rules (Definition 8)

$$\frac{}{\Delta \vdash a[a \mapsto t] \to t}\ (\beta\mathbf{a}) \qquad \frac{\Delta \vdash a\#r}{\Delta \vdash r[a \mapsto t] \to r}\ (\beta\#) \qquad \frac{}{\Delta \vdash (\lambda X.r)t \to r[X := t]}\ (\beta\mathbf{2})$$

$$\frac{\Delta \vdash a\#r}{\Delta \vdash (r'r)[a \mapsto t] \to (r'[a \mapsto t])r}\ (\beta\mathbf{2app}) \qquad \frac{level(r') = 1}{\Delta \vdash (r'r)[a \mapsto t] \to (r'[a \mapsto t])(r[a \mapsto t])}\ (\beta\mathbf{1app})$$

$$\frac{\Delta \vdash b\#t}{\Delta \vdash (\lambda b.r)[a \mapsto t] \to \lambda b.(r[a \mapsto t])}\ (\beta\lambda\mathbf{1}) \qquad \frac{(X \notin fv(t))}{\Delta \vdash (\lambda X.r)[a \mapsto t] \to \lambda X.(r[a \mapsto t])}\ (\beta\lambda\mathbf{2})$$

Figure 4. Reductions of two-and-a-halfth order λ-calculus (Definition 11)

$$\frac{}{\Delta \vdash a \Rightarrow a}\ (\mathbf{Pa}) \qquad \frac{}{\Delta \vdash (\pi\cdot X) \Rightarrow (\pi\cdot X)}\ (\mathbf{PX}) \qquad \frac{\Delta \vdash r \Rightarrow s \quad \Delta \vdash t \Rightarrow u}{\Delta \vdash rt \Rightarrow su}\ (\mathbf{Papp})$$

$$\frac{\Delta \vdash r \Rightarrow s}{\Delta \vdash \lambda a.r \Rightarrow \lambda a.s}\ (\mathbf{P}\lambda\mathbf{a}) \qquad \frac{\Delta \vdash r \Rightarrow s}{\Delta \vdash \lambda X.r \Rightarrow \lambda X.s}\ (\mathbf{P}\lambda\mathbf{X})$$

$$\frac{\Delta \vdash r \Rightarrow s \quad \Delta \vdash t \Rightarrow u \quad \Delta \vdash su \xrightarrow{R} v}{\Delta \vdash rt \Rightarrow v}\ (\mathbf{Papp}\varepsilon) \qquad \frac{\Delta \vdash r \Rightarrow s \quad \Delta \vdash a\#s \quad \Delta \vdash b\#s}{\Delta \vdash r \Rightarrow (a\ b)\cdot s}\ (\mathbf{P}\alpha)$$

Figure 5. The parallel reduction relation (Definition 35)

tution is as usual within a single level:

$$(\lambda b.(\lambda a.b))a \to \lambda a'.(b[b \mapsto a]) \to \lambda a'.a$$
$$(\lambda Y.(\lambda X.Y))X \to \lambda X'.(Y[Y := X]) \equiv \lambda X'.X$$

Remark 12 (β**1app**) and (β**2app**) can be viewed as two parts of a single rule; if $level(r) = 1$ *and* $\Delta \vdash b\#t$ we may join (β**1app**) and (β**2app**) with ($\beta\#$).

Suppose we allow $(r'r)[b \mapsto u] \overset{(\textbf{FALSE})}{\longrightarrow} (r'[b \mapsto u])(r[b \mapsto u])$. Then:

$$((\lambda X.(a\ b) \cdot X)b)[b \mapsto c] \to ((a\ b) \cdot X)[X := b][b \mapsto c] \equiv a[b \mapsto c] \to a$$
$$((\lambda X.(a\ b) \cdot X)b)[b \mapsto c] \to (\lambda X.(a\ b) \cdot X)[b \mapsto c](b[b \mapsto c]) \to \cdots \to c$$

a and c cannot be joined. So, without conditions confluence fails.

To close a divergence in $((\lambda X.r)t)[b \mapsto u]$ between (β**2**) and (**FALSE**) we must join $r[X := t][b \mapsto u]$ and $r[b \mapsto u][X := t[b \mapsto u]]$ where $X \notin fv(u)$ and $X \notin fv(t)$ and where the freshness context Δ is such that $\Delta \vdash b\#u$. It is not possible to join this in general — take $r \equiv (a\ b) \cdot X$, $t \equiv a$, and $u \equiv c$. However, it is possible to join this if $level(r) = 1$ or if $\Delta \vdash b\#t$.

In the rest of this section we prove soundness results; that freshness and reduction are preserved under instantiating unknowns (Lemma 14 and Theorem 18), and that freshness is preserved under reduction (Theorem 16):

Definition 13 A **substitution** σ is a finitely supported map from level 2 variables to terms. 'Finitely supported' means that $\sigma(X) \equiv \mathrm{id} \cdot X$ for all but finitely many variables. These act on terms $t\sigma$ in the natural way extending the action of $[X := t]$ from Definition 2. We extend this action pointwise as convenient; in particular we write $\Delta\sigma$ for $\{a\#\sigma(X) \mid a\#X \in \Delta\}$.

If $\mathcal{F}$ is a set of freshnesses write $\Delta' \vdash \mathcal{F}$ for '$\Delta' \vdash a\#r$ for every $a\#r \in \mathcal{F}$'. Lemma 14 is soundness for substitutions compatible with the freshness context:

Lemma 14 *If* $\Delta' \vdash \Delta\sigma$ *then* $\Delta \vdash a\#r$ *implies* $\Delta' \vdash a\#(r\sigma)$.

Proof By induction on r. We consider two cases:

– The case of λa. $\Delta' \vdash a\#(\lambda a.r)\sigma$ always, by ($\mathbf{a\#\lambda a}$).

– The case of λX. Suppose $\Delta, a\#X \vdash \pi(a)\#\pi \cdot r$ is derivable, where $X \notin \Delta$. Suppose the inductive hypothesis of the derivation. Renaming X if necessary we may assume that $X \notin \Delta'$ and $\sigma(X) \equiv X$. By inductive hypothesis $\Delta', a\#X \vdash \pi(a)\#(\pi \cdot r)\sigma$ is derivable. By Lemma 5, $(\pi \cdot r)\sigma \equiv \pi \cdot (r\sigma)$. We extend the derivation with ($\mathbf{a\#\lambda X}$) to obtain a derivation of $\Delta' \vdash \pi(a)\#\pi \cdot (\lambda X.(r\sigma))$. We then have $\pi \cdot (\lambda X.(r\sigma)) \equiv \pi \cdot ((\lambda X.r)\sigma) \equiv (\pi \cdot \lambda X.r)\sigma$ (the final $\equiv$ uses Lemma 5) and the result follows.

□

Lemma 15 $\Delta \vdash a\#r$ *implies* $\Delta \vdash \pi(a)\#\pi \cdot r$.

Proof By induction on derivations, we consider just one case:

– The case ($\mathbf{a\#\lambda X}$). Suppose $\Delta, a\#X \vdash \pi(a)\#\pi \cdot r$. By inductive hypothesis $\Delta, a\#X \vdash \pi'(\pi(a))\#\pi' \cdot (\pi \cdot r)$, so $\Delta, a\#X \vdash (\pi' \circ \pi)(a)\#(\pi' \circ \pi) \cdot r$ by Lemma 4.

Using ($\mathbf{a}\#\lambda\mathbf{X}$) we derive $\Delta \vdash (\pi' \circ \pi)(a)\#(\pi' \circ \pi) \cdot \lambda X.r$. We use Lemma 4 to deduce $\Delta \vdash \pi'(\pi(a))\#\pi' \cdot (\pi \cdot \lambda X.r)$.

□

Theorem 16 is 'subject-reduction for freshness':

Theorem 16 *If* $\Delta \vdash r \to s$ *then* $\Delta \vdash a\#r$ *implies* $\Delta \vdash a\#s$.

Proof By induction on derivations. We consider a selection of cases:

– The case (β**2app**), assuming $\Delta \vdash b\#r$. Suppose $\Delta \vdash b\#r$ and $\Delta \vdash (r'r)[b \mapsto u] \to (r'[b \mapsto u])r$.

If $\Delta \vdash a\#(r'r)[b \mapsto u]$ then $\Delta \vdash a\#r'$, $\Delta \vdash r$, and $\Delta \vdash u$. It follows that $\Delta \vdash a\#(r'[b \mapsto u])r$. Similarly if $\Delta \vdash b\#(r'r)[b \mapsto u]$. Similarly for ($\beta$**1app**).

– The case (β**2**). Suppose that $\Delta \vdash (\lambda X.r)t \to r[X := t]$.

If $\Delta \vdash a\#(\lambda X.r)t$ then $\Delta \vdash a\#t$ and $\Delta, a\#X \vdash a\#r$. It follows by Lemma 14 that $\Delta \vdash a\#r[X := t]$ as required.

– The case ($\triangleright\alpha$). Suppose that $\Delta \vdash r \to s$ and $\Delta \vdash a\#s$ and $\Delta \vdash b\#s$.

$\Delta \vdash a\#s$ and $\Delta \vdash b\#s$ by assumption. If $\Delta \vdash c\#r$ then by inductive hypothesis $\Delta \vdash c\#s$, and so $\Delta \vdash c\#(a\ b) \cdot s$ by Lemma 15.

□

Lemma 17 *If* $level(r) = 1$ *then* $level(r\sigma) = 1$.

Theorem 18 does for $\to$ what Lemma 14 did for freshness; it is a form of soundness under substitutions compatible with the freshness contexts:

Theorem 18 *If* $\Delta' \vdash \Delta\sigma$ *then* $\Delta \vdash r \to s$ *implies* $\Delta' \vdash r\sigma \to s\sigma$.

Proof By induction on derivations. We present a selection of cases.

– The case ($\beta\#$). Suppose $\Delta \vdash a\#r$. By Theorem 16 $\Delta' \vdash a\#r\sigma$ if $\Delta \vdash a\#r$. It follows by ($\beta\#$) that $\Delta' \vdash r[a \mapsto t] \to r$.

– The case (β**1app**). We have $(r'r)[a \mapsto t]\sigma \equiv (r'\sigma)(r\sigma)[a \mapsto t\sigma]$. By assumption $level(r') = 1$ so by Lemma 17, $level(r'\sigma) = 1$ and $\Delta \vdash (r'\sigma)(r\sigma)[a \mapsto t] \to (r'[a \mapsto t])(r[a \mapsto t])\sigma$ by (β**1app**). The result follows.

– The case ($\beta\lambda$**1**). By Theorem 16 if $\Delta \vdash b\#t$ then $\Delta' \vdash b\#t\sigma$. It follows by ($\beta\lambda$**1**) that $\Delta' \vdash (\lambda b.r\sigma)[a \mapsto t\sigma] \to \lambda b.(r\sigma[a \mapsto t\sigma])$.

□

4 Confluence

Our calculus can be viewed as two λ-calculi (level 1, level 2) glued together by a nominal treatment of the interaction of level 1 α-equivalence with level 2 variables. The proof of confluence also splits in two proofs, plus some 'proof-glue'.

4.1 Level 1 reductions

Definition 19 Let (**level1**) be the set $\{(\beta\mathbf{a}), (\beta\#), (\beta\mathbf{1app}), (\beta\mathbf{2app}), (\beta\lambda\mathbf{1}), (\beta\lambda\mathbf{2})\}$ of rules from Figure 4. Let $\Delta \vdash r \overset{(\mathbf{level1})}{\longrightarrow} s$ be the least congruence closed under the rules in (**level1**).

Definition 20 Say that Δ^+ **freshly extends** Δ when $\Delta^+ = \Delta \cup \Delta'$ where for all $c\#X \in \Delta'$, $c \notin \Delta$ and $X \in \Delta$ (so Δ^+ 'extends Δ with some fresh atoms').

Definition 21 Write $\Delta \nvdash a\#r$ when $\Delta \vdash a\#r$ is *not* derivable.

Choose some fixed but arbitrary order on atoms. If S is a finite set of atoms say 'for the first atom not in S' to mean 'for the least atom, in our fixed but arbitrary order, that is not an element of S'. [1]

Definition 22 For a given Δ, define a **level 1 substitution action** $r[a := t]$ as below; earlier rules take priority.

- $r[a := t] \equiv r$ if $\Delta \vdash a\#r$.
- $a[a := t] \equiv t$.
- $(\pi \cdot X)[a := t] \equiv (\pi \cdot X)[a \mapsto t]$.
- $(r'r)[a := t] \equiv (r'[a := t])(r[a := t])$ provided $level(r') = 1$.
- $(r'r)[a := t] \equiv r'[a := t]r$ provided $\Delta \vdash a\#r$.
- $(r'r)[a := t] \equiv (r'r)[a \mapsto t]$ if $\Delta \nvdash a\#r$ and $level(r') = 2$.
- $(\lambda b.r)[a := t] \equiv \lambda b.(r[a := t])$ provided $\Delta \vdash b\#t$.
- $(\lambda b.r)[a := t] \equiv (\lambda c.((b\ c) \cdot r)[a := t])$ if $\Delta \nvdash b\#t$. Here c is the first atom not mentioned in r, a, b, or t, such that $\Delta \vdash c\#r$ and $\Delta \vdash c\#t$, if such a c exists. Thus we α-convert the level 1 λb to a fresh λc, but in the presence of level 2 variables we use an explicit permutation and explicitly fresh level 1 variable, in nominal terms style.
- $(\lambda b.r)[a := t] \equiv (\lambda b.r)[a \mapsto t]$ otherwise.
- $(\lambda b.r)[a := t] \equiv (\lambda b.r)[a := t]$ if $\Delta \nvdash b\#t$.
- $(\lambda X.r)[a := t] \equiv \lambda X.(r[a := t])$, renaming X if needed so that $X \notin t$.

Then for that Δ, let r^* be defined as follows; earlier rules have priority, read left-to-right then top-to-bottom (Δ will always be clear from the context):

$$a^* \equiv a \quad (\pi \cdot X)^* \equiv \pi \cdot X \quad (\lambda a.r)^* \equiv \lambda a.r^* \quad (\lambda X.r)^* \equiv \lambda X.r^*\ (X \notin \Delta)$$
$$(r'[a \mapsto t])^* \equiv r'^*[a := t^*] \quad ((\lambda X.r)t)^* \equiv r^*[X := t^*] \quad (r'r)^* \equiv r'^*r^*$$

We can always rename X to ensure $X \notin \Delta$.

- If $\Delta = \{a\#X\}$ then $X[a := b] \equiv X$ and $(X[a := b])^* \equiv X$.
- If $\Delta = \{\}$ then $X[a := b] \equiv X[a \mapsto b]$.

Intuitively, r^* is a canonical form of r and $r[a := t]$ one of $r[a \mapsto t]$. This is not necessarily a normal form (which may not exist) but garbage is collected and substitutions (level 1 β-reducts) are pushed into r. Level 2 β-reducts are unreduced.

Definition 23 Call $\Delta \vdash \text{-} \triangleright \text{-}$ **reflexive** when $\Delta \vdash r \triangleright r$ always, and **transitive** when $\Delta \vdash r \triangleright r'$ and $\Delta \vdash r' \triangleright r''$ imply $\Delta \vdash r \triangleright r''$. Write $\Delta \vdash \text{-} \triangleright^* \text{-}$ for the least transitive reflexive relation containing $\Delta \vdash \text{-} \triangleright \text{-}$.

Definition 24 Suppose that Δ and Δ^+ are freshness contexts. Say Δ^+ **freshly extends** Δ when there exists some Δ' such that: $\Delta \cap \Delta' = \emptyset$ (in words: Δ and Δ'

[1] This is not necessary for expressing the proofs to follow but it is convenient. We will never make infinitely many choices of fresh atom, and nowhere will the truth of a result depend on our choice of order.

are disjoint), $\Delta^+ = \Delta \cup \Delta'$, for all $c\#X \in \Delta'$ it is the case that $c \notin \Delta$, and for all $c\#X \in \Delta'$ it is the case that $X \in \Delta$.

Intuitively, Δ^+ 'extends Δ with some fresh atoms'.

Lemma 25 *For every Δ, r, a, and t there exists a Δ^+ freshly extending Δ such that $\Delta^+ \vdash r[a \mapsto t] \overset{(\textbf{level1})}{\longrightarrow}{}^* r[a := t]$ ($r[a := t]$ calculated for Δ^+.)*

Proof Each rule in Definition 22 is emulated by a rule in (**level1**). We may need some fresh atoms to α-convert. □

Lemma 26 *$depth(r) = depth(r[X := \pi \cdot Y])$*

Lemma 27 (i) $(\pi \cdot r)^* \equiv \pi \cdot r^*$

(ii) *If $(\pi \cdot t)^* \equiv \pi \cdot t^*$ for all π, then $(r[X := t])^* \equiv r^*[X := t^*]$*

Proof By induction on $depth(r)$, using Lemma 6 and Lemma 26. □

Theorem 28 states that terms reduce to their canonical form; it enters via Lemma 25 that we may need some fresh atoms, to α-convert.

Theorem 28 *For every Δ and r there exists a Δ^+ freshly extending Δ such that $\Delta^+ \vdash r \overset{(\textbf{level1})}{\longrightarrow}{}^* r^*$ (r^* calculated for Δ^+.)*

Proof By induction on r. For example, $a^* \equiv a$ and $(\lambda X.r)^* \equiv \lambda X.r^*$. The special case $(\lambda X.r)t$ requires Lemma 27, and the case $(\lambda a.r)t$ requires Lemma 25. □

Lemma 29 *Fix Δ. Then $\Delta \vdash a\#s$ implies $\Delta \vdash a\#s^*$. (s^* calculated for Δ.)*

Proof By induction on derivations, using Lemma 27. □

Lemma 30 *If $\Delta \vdash r \overset{(\textbf{level1})}{\longrightarrow} s$ and $\Delta' \vdash \Delta[X := \pi \cdot X]$, then $\Delta' \vdash r[X := \pi \cdot X] \overset{(\textbf{level1})}{\longrightarrow} s[X := \pi \cdot X]$.*

Proof By induction on derivations, using Lemma 26, Lemma 14 and Lemma 5. □

Lemma 31 *If $\Delta \vdash r \overset{(\textbf{level1})}{\longrightarrow} s$ then $\Delta \vdash \pi \cdot r \overset{(\textbf{level1})}{\longrightarrow} \pi \cdot s$.*

Proof By induction on derivations, using Lemma 15 and Lemma 30. □

Theorem 32, along with Theorem 28 above, makes up the diagram in Theorem 33:

Theorem 32 *For every Δ, r, and s, there exists a Δ^+ freshly extending Δ such that $\Delta \vdash r \overset{(\textbf{level1})}{\longrightarrow} s$ implies $\Delta^+ \vdash s^* \overset{(\textbf{level1})}{\longrightarrow}{}^* r^*$. ($r^*$ and s^* calculated for Δ^+.)*

Proof By induction on the derivation of $\Delta \vdash r \overset{(\textbf{level1})}{\longrightarrow} s$. We sketch some cases:

– The case (β**2app**) where $\Delta \vdash a\#r$. We use Lemma 29 to conclude that $\Delta^+ \vdash a\#r^*$ in the case where $level(r') = 1$.

$$\begin{array}{lll} \Delta \vdash (r'r)[a{\mapsto}t] \to (r'[a{\mapsto}t])r & & \\ ((r'r)[a{\mapsto}t])^* \equiv (r'^*[a{:=}t^*])(r^*[a{:=}t^*]) \equiv (r'^*[a{:=}t^*])(r^*) & \text{if } level(r') = 1 \\ ((r'r)[a{\mapsto}t])^* \equiv (r'^*[a{:=}t^*])(r^*) & \text{if } level(r') = 2 \end{array}$$

– The case (β**1app**) where $level(r') = 1$.

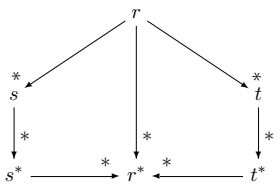
$$\Delta \vdash (r'r)[a \mapsto t] \to (r'[a \mapsto t])(r[a \mapsto t])$$
$$((r'[a \mapsto t])(r[a \mapsto t]))^* \equiv (r'^*[a := t^*])(r^*[a := t^*]) \equiv ((r'r)[a \mapsto t])^*$$

– The case ($\beta\lambda$**2**). Suppose $X \notin t$, which can be guaranteed. Then $\Delta \vdash (\lambda X.r)[a \mapsto t] \overset{(\textbf{level1})}{\longrightarrow} \lambda X.(r[a \mapsto t])$. We have $((\lambda X.r)[a \mapsto t])^* \equiv \lambda X.r^*[a := t^*] \equiv (\lambda X.(r[a \mapsto t]))^*$, and we have the result.
– The case ($\triangleright\alpha$). Suppose $s \equiv (a\ b) \cdot s'$. Suppose $\Delta \vdash r \to s'$ and $\Delta \vdash a\#s'$ and $\Delta \vdash b\#s'$. By inductive hypothesis, $\Delta^+ \vdash s'^* \overset{(\textbf{level1})}{\longrightarrow}{}^* r^*$. By Lemma 29 $\Delta^+ \vdash a\#s'^*$ and $\Delta^+ \vdash b\#s'^*$. By Theorem 16 it follows that $\Delta^+ \vdash a\#r^*$ and $\Delta^+ \vdash b\#r^*$. By ($\triangleright\alpha$) it follows that $\Delta^+ \vdash s'^* \overset{(\textbf{level1})}{\longrightarrow}{}^* (a\ b) \cdot (r^*)$. By Lemmas 31 and 4 it follows that $\Delta^+ \vdash (a\ b) \cdot (s'^*) \overset{(\textbf{level1})}{\longrightarrow}{}^* r^*$. By Lemma 27 $\Delta^+ \vdash s^* \overset{(\textbf{level1})}{\longrightarrow}{}^* r^*$ as required.

□

Theorem 33 (**level1**) *is confluent.*

Proof From Theorem 28 and Theorem 32: □



4.2 Level 2

Definition 34 Let (**level2**) be the set $\{(\beta\mathbf{2}), (\beta\lambda\mathbf{2})\}$ of rules from Figure 4. Let $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow} s$ be the least congruence closed under the rules in (**level2**).

Note that $\overset{(\textbf{level1})}{\longrightarrow}$ and $\overset{(\textbf{level2})}{\longrightarrow}$ have ($\beta\lambda$**2**) in common. This is necessary for confluence to work, see Lemma 46.

Definition 35 Define a **parallel reduction relation**, $\Rightarrow$, by the rules in Figure 5. Here, R ranges over rules in (**level2**) (Definition 34).

Lemma 36 $\Delta \vdash r \Rightarrow s$ *implies* $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow}{}^* s$. *As a corollary,* $\Delta \vdash r \Rightarrow^* s$ *implies* $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow}{}^* s$.

Lemma 37 $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow} s$ *implies* $\Delta \vdash r \Rightarrow s$. *As a corollary,* $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow}{}^* s$ *implies* $\Delta \vdash r \Rightarrow^* s$

Proof It suffices to show that every possible (**level2**)-reduction can be mirrored by a parallel reduction. This is routine. The second part follows from the first part and an easy proof that $\Delta \vdash r \Rightarrow r$ ($\Rightarrow$ is reflexive). □

Theorem 38 $\Delta \vdash r \Rightarrow^* s$ *if and only if* $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow}{}^* s$

Proof From Lemmas 36 and 37. □

Lemma 39 *If* $\Delta \vdash r \overset{(\textbf{level2})}{\longrightarrow} s$ *then* $\Delta \vdash \pi \cdot r \overset{(\textbf{level2})}{\longrightarrow} \pi \cdot s$.

Proof By induction on derivations. We mention two cases:

– The case ($\triangleright\alpha$). By inductive hypothesis, we have $\Delta \vdash \pi \cdot r \to \pi \cdot s$ and from Lemma 15 we have $\Delta \vdash \pi(a)\#\pi \cdot s$ and $\Delta \vdash \pi(b)\#\pi \cdot s$. Using ($\triangleright\alpha$) we obtain $\Delta \vdash (\pi \cdot r) \to (\pi(a)\ \pi(b)) \cdot (\pi \cdot s)$ and the result follows via elementary properties of permutations.

– The case ($\beta\mathbf{2}$). Using Lemma 3, we have:

$$\begin{array}{rcl}\pi\cdot((\lambda X.r)t) & \equiv & (\lambda X.\pi\cdot r[X{:=}\pi^{-1}\cdot X])(\pi\cdot t) \;\rightarrow\; (\pi\cdot r[X{:=}\pi^{-1}\cdot X])[X{:=}\pi\cdot t] \\ & \equiv & \pi\cdot(r[X{:=}\pi^{-1}\cdot X][X{:=}\pi\cdot t]) \;\equiv\; (\pi\cdot r)[X{:=}t] \;\equiv\; \pi\cdot(r[X{:=}t])\end{array}$$

□

Lemma 40 *If* $\Delta \vdash r{\Rightarrow}s$ *and* $\Delta' \vdash \Delta[X{:=}\pi\cdot X]$ *then* $\Delta' \vdash r[X{:=}\pi\cdot X]{\Rightarrow}s[X{:=}\pi\cdot X]$.

Proof By induction on derivations using Lemma 5, Theorem 18 and Lemma 14.□

Lemma 41 *If* $\Delta \vdash r \Rightarrow s$ *then* $\Delta \vdash \pi \cdot r \Rightarrow \pi \cdot s$.

Proof The proof is by induction on derivations. We present a selection of cases:

– The case ($\mathbf{P}\lambda\mathbf{X}$). By inductive hypothesis, we have $\Delta \vdash \pi \cdot r \Rightarrow \pi \cdot s$. From this and Lemma 40 we can obtain $\Delta \vdash (\pi \cdot r)[X := \pi^{-1} \cdot X] \Rightarrow (\pi \cdot s)[X := \pi^{-1} \cdot X]$. Using ($\mathbf{P}\lambda\mathbf{X}$), we have $\Delta \vdash \pi \cdot \lambda X.r \Rightarrow \pi \cdot \lambda X.s$.
– The case ($\mathbf{Papp}\varepsilon$). By inductive hypothesis, we have $\Delta \vdash \pi \cdot r \Rightarrow \pi \cdot s$ and $\Delta \vdash \pi \cdot t \Rightarrow \pi \cdot u$ and that $\Delta \vdash (\pi \cdot s)(\pi \cdot u) \overset{(\mathbf{level2})}{\longrightarrow} \pi \cdot v$, using Lemma 39. Then $\Delta \vdash (\pi \cdot r)(\pi \cdot t) \Rightarrow \pi \cdot v$ is derivable, and the result follows.
– The case ($\mathbf{P}\alpha$). By inductive hypothesis, $\Delta \vdash \pi \cdot r \Rightarrow \pi \cdot s$, and by Lemma 15, $\Delta \vdash \pi(a)\#\pi \cdot s$ and $\Delta \vdash \pi(b)\#\pi \cdot s$. Then $\Delta \vdash \pi \cdot r \Rightarrow (\pi(a)\ \pi(b)) \cdot (\pi \cdot s)$ is derivable and the result follows by elementary properties of permutations.

□

Lemma 42 $\Delta \vdash r \Rightarrow s$ *and* $\Delta \vdash t \Rightarrow u$ *imply* $\Delta \vdash r[X := t] \Rightarrow s[X := u]$.

Proof The proof is by induction on the derivation of $\Delta \vdash r \Rightarrow s$ (Figure 5). We assume that $\Delta \vdash t \Rightarrow u$.

– The case ($\mathbf{PX}$). By Lemma 41, and by the fact that $(\pi \cdot Y)[X := t] \equiv \pi \cdot Y$.
– The case ($\mathbf{Papp}\varepsilon$). Suppose $\Delta \vdash r \Rightarrow s$ and $\Delta \vdash v \Rightarrow w$ and $\Delta \vdash sw \overset{(\mathbf{level2})}{\longrightarrow} v$. Then $\Delta \vdash r[X := t] \Rightarrow s[X := u]$ and $\Delta \vdash v[X := t] \Rightarrow w[X := u]$ are both derivable, and from Theorem 18, we have $\Delta \vdash s[X := u](w[X := u]) \rightarrow v[X := u]$. The result follows from ($\mathbf{Papp}\varepsilon$).
– The case ($\mathbf{P}\alpha$). By inductive hypothesis, $\Delta \vdash r[X := t] \Rightarrow s[X := u]$, and by Lemma 14, $\Delta \vdash a\#s[X := u]$ and $\Delta \vdash b\#s[X := u]$. Extending with ($\mathbf{P}\alpha$), we obtain $\Delta \vdash r[X := t] \Rightarrow (a\ b) \cdot (s[X := t])$. The result follows from Lemma 5.

□

Write '$\Delta \vdash r \Leftarrow s \Rightarrow t$' for '$\Delta \vdash r \Rightarrow s$ and $\Delta \vdash r \Rightarrow t$', and similarly for other reduction relations later.

Lemma 43 $\Delta \vdash$ - $\Rightarrow$ - *satisfies the diamond property: If* $\Delta \vdash r \Leftarrow s \Rightarrow t$ *then there exists some* u *such that* $\Delta \vdash r \Rightarrow u \Leftarrow t$.

Proof We consider possible pairs of rules that can derive $\Delta \vdash s \Rightarrow r$ and $\Delta \vdash s \Rightarrow t$, where $r \not\equiv t$. We present the case of ($\mathbf{Papp}$) and ($\mathbf{Papp}\varepsilon$) for ($\beta\mathbf{2}$). Suppose $\Delta \vdash (\lambda X.r')t' \Leftarrow (\lambda X.r)t \Rightarrow r''[X := t'']$ using ($\mathbf{Papp}$), and ($\mathbf{Papp}\varepsilon$) for ($\beta\mathbf{2}$). By inductive hypothesis, there exists an r''' such that $\Delta \vdash r' \Rightarrow r''' \Leftarrow r''$ and a t'''

such that $\Delta \vdash t' \Rightarrow t''' \Leftarrow t''$. Using Lemma 42 we derive $\Delta \vdash (\lambda X.r')t' \Rightarrow r'''[X := t'''] \Leftarrow r''[X := t'']$. □

Theorem 44 $\overset{(\textbf{level2})}{\longrightarrow}$ *is confluent.*

Proof An immediate corollary of Lemma 43 and Theorem 38. □

4.3 Level 1 and level 2 reductions

Lemma 45 $\Delta \vdash r \to s$ *implies* $level(s) \leq level(r)$*, and so does* $\Delta \vdash r \Rightarrow s$*.*

Lemma 46 $\Delta \vdash r \Leftarrow s \overset{(\textbf{level1})}{\longrightarrow} t$ *implies that* $\Delta \vdash r \overset{(\textbf{level1})}{\longrightarrow}{}^{*} u \Leftarrow t$*. As a corollary,* $\Delta \vdash r \Leftarrow s \overset{(\textbf{level1})}{\longrightarrow}{}^{*} t$ *implies* $\Delta \vdash r \overset{(\textbf{level1})}{\longrightarrow}{}^{*} u \Leftarrow t$*.*

Proof We assume $\Delta \vdash u \Rightarrow u'$, $\Delta \vdash t \Rightarrow t'$, and so on. We present some cases:
$\Delta \vdash (r'''r'')[a \mapsto t'] \Leftarrow (r'r)[a \mapsto t] \to (r'[a \mapsto t])(r[a \mapsto t])$ when $level(r') = 1$.
This can be closed to $(r'''[a \mapsto t'])(r''[a \mapsto t'])$; by Lemma 45 $level(r''') = 1$.
$\Delta \vdash (r'''r'')[a \mapsto t'] \Leftarrow (r'r)[a \mapsto t] \to (r'[a \mapsto t])r$ when $\Delta \vdash a\#r$.
This can be closed to $(r'''[a \mapsto t'])r''$.
$\Delta \vdash ((\lambda X.r')t')[b \mapsto u'] \Leftarrow ((\lambda X.r)t)[b \mapsto u] \to ((\lambda X.r)[b \mapsto u])t$ when $\Delta \vdash b\#t$.
This can be closed to $((\lambda X.r')[b \mapsto u'])t'$.
$\Delta \vdash r'[X := t'][b \mapsto u'] \Leftarrow ((\lambda X.r)t)[b \mapsto u] \to ((\lambda X.r)[b \mapsto u])t$ when $\Delta \vdash b\#t$.
This can be closed to $r'[X := t'][b \mapsto u']$ where $X \notin u$ which can be guaranteed; we require $(\beta\lambda\mathbf{2})$ in (**level2**).

□

Lemma 47 *If* $\Delta \vdash r \overset{(\textbf{level1})}{\longleftarrow} s \to^{*} t$*, then* $\Delta^{+} \vdash r \to^{*} u \longleftarrow t$ *for some suitably freshened context,* Δ^{+}*. Similarly, if* $\Delta \vdash r \overset{(\textbf{level2})}{\longleftarrow} s \to^{*} t$*, then* $\Delta^{+} \vdash r \to^{*} u \longleftarrow t$ *for some suitable freshened context* Δ^{+}*.*

Proof Both claims follow by induction on the path length of $\Delta \vdash s \to^{*} t$ using Theorem 33 and Theorem 44 and Lemma 46. □

Theorem 48 $\Delta \vdash \text{-} \to \text{-}$ *(reduction with* (**level2**) ∪ (**level1**)*) is confluent, in the following sense: if* $\Delta \vdash r \to^{*} s$ *and* $\Delta \vdash r \to^{*} s'$ *then there exists some* Δ^{+} *freshly extending* Δ*, and some* u*, such that* $\Delta^{+} \vdash s \to^{*} u$ *and* $\Delta^{+} \vdash s' \to^{*} u$*.*

Proof By a diagrammatic argument, using Lemma 47. □

The 'freshly extending' part of Theorem 48 ensures we can α-rename λa with a permutation to guarantee the freshness precondition of $(\beta\lambda\mathbf{1})$. In the presence of only a single level of variable we can 'just rename', and so we do not need to record the freshness of freshly generated variables relative to higher-level variables in a freshness context.

Recall Definition 23. Call $\Delta \vdash \text{-} \triangleright \text{-}$ **symmetric** when $\Delta \vdash r \triangleright r'$ implies $\Delta \vdash r' \triangleright r$. Write $\Delta \vdash \text{-} = \text{-}$ for the least congruence closed under $\Delta \vdash \text{-} \to \text{-}$ that is also transitive, reflexive, and symmetric.

Corollary 49 $\Delta \vdash \text{-} = \text{-}$ *is consistent. (There exist two terms which are not related by the transitive reflexive symmetric closure of* $\Delta \vdash \text{-} \to \text{-}$*.)*

Proof $\lambda a.\lambda b.a$ and $\lambda a.\lambda b.b$ are two distinct terms, and they do not rewrite to a common term. By Theorem 48 the result follows. □

5 Related and future work

Previous 'nominal' λ-calculi.

The NEW calculus of contexts and the λ-context calculus [6,8] are λ-calculi with more than one level of variable. These have a weak theory of α-equivalence far less powerful than that of nominal terms (they also have an infinite hierarchy of levels; combining this with nominal terms style α-equivalence is future work).

The lambda context calculus preserves strong normalisation. A variant of two-and-a-halfth order λ-calculus which preserves strong normalisation is possible using ideas from [8]. The proof of confluence is not harder. The result in this paper is more relevant for designing logics, where we trade off more reductions against weaker computational properties.

Capture-avoiding substitution as a nominal algebra with Mathijssen [9,12] used nominal terms syntax, with permutations and freshness; substitution corresponds to level 1 β-reducts. There is no level 2 λ-abstraction $\lambda X.$

λ-calculi for capturing substitution.

The λ-calculus itself can emulate capturing substitution. Let r and t range over λ-terms: then the quote of the Introduction is emulated by 'apply $\lambda x.((\lambda y.r)(yx))$ to $\lambda x.t$'. However, compositionality fails in the sense that the number of 'extra' λ-abstractions needed on t, and 'extra' applications needed on y, depend non-locally on the variables for which we authorise capture at the point(s) at which y occurs in r. See [17, Section 2] for further discussion.

λ-calculi exist with more than one level of variable, or with constructions with essentially the same intent; they are broadly called 'calculi of contexts'. Calculi include 'holes with binding power' by Jojgov and Geuvers [15], two calculi of contexts λm and $\lambda\mathcal{M}$ by Sato and others [19], further work also by Sato and others [18,14], and λc by Bognar [1, Section 2]. Two-and-a-halfth-order λ-calculus is specifically tailored to nominal terms and so fits into the authors' own research programme, and it may offer other advantages too: a relatively simple syntax and the possibility to import other nominal research, from semantics [13,7] to implementation [21]. We have not yet built semantics or a type system for two-and-a-halfth order λ-calculus — but tools exist to do this in a principled way [7,4].

A thread of research exemplified by [17] manages capturing vs. non-capturing substitution inside a specially-designed logical framework. Our calculus uses levels; there is no (need for a) 'wrapping' in formal logic. Also, our level 2 variables are 'open' (instantiation captures unless a freshness condition forbids it) whereas context variables in [17] are 'closed' (variables that can be captured must be accounted for, in the type system).

Future work.

We intend to impose types and create a higher-order logic (this might have elements in common with the logical framework of [17]; a discussion will follow once

the logic is created). Nominal unification, nominal algebra, and one-and-a-halfth order logic [20,16,11] have already done much of the work but the universal quantification of unknowns X is implicit (there is no λX or $\forall X$). Denotations, presumably using nominal sets [13], are future work. We can also consider enriching our functional operational semantics with nominal unification, which is computationally more tractable than higher-order unification [20].

References

[1] Mirna Bognar. *Contexts in Lambda Calculus*. PhD thesis, Vrije Universiteit Amsterdam, 2002.

[2] James Cheney. Nominal logic and abstract syntax. *SIGACT News (logic column 14)*, 36(4):47–69, 2005.

[3] James Cheney and Christian Urban. Alpha-prolog: A logic programming language with names, binding and alpha-equivalence. In Bart Demoen and Vladimir Lifschitz, editors, *Proc. of the 20th Int'l Conf. on Logic Programming (ICLP 2004)*, number 3132 in LNCS, pages 269–283. Springer-Verlag, 2004.

[4] Maribel Fernández and Murdoch J. Gabbay. Curry-style types for nominal rewriting. TYPES'06, 2006.

[5] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting. *Information and Computation*, 205(6):917–965, 2007.

[6] Murdoch J. Gabbay. A NEW calculus of contexts. In *PPDP'05*, pages 94–105. ACM, 2005.

[7] Murdoch J. Gabbay. A General Mathematics of Names. *Information and Computation*, 205:982–1011, July 2007.

[8] Murdoch J. Gabbay and Stéphane Lengrand. The lambda-context calculus. *ENTCS*, 196:19–35, 2008.

[9] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding Substitution as a Nominal Algebra. In *ICTAC*, volume 4281 of *LNCS*, pages 198–212, 2006.

[10] Murdoch J. Gabbay and Aad Mathijssen. A formal calculus for informal equality with binding. In *WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation*, volume 4576 of *LNCS*, pages 162–176, 2007.

[11] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic (journal version). *Journal of Logic and Computation*, November 2007. Online.

[12] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding Substitution as a Nominal Algebra (journal version). *Formal Aspects of Computing*, 2008. Online.

[13] Murdoch J. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax with Variable Binding (journal version). *Formal Aspects of Computing*, 13(3–5):341–363, 2001.

[14] Masatomo Hashimoto and Atsushi Ohori. A typed context calculus. *Theoretical Computer Science*, 266(1-2):249–272, 2001.

[15] Gueorgui I. Jojgov. Holes with binding power. In *TYPES*, volume 2646 of *LNCS*, pages 162–181. Springer, 2002.

[16] Aad Mathijssen. *Logical Calculi for Reasoning with Binding*. PhD thesis, Technische Universiteit Eindhoven, 2007.

[17] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *Transactions on Computational Logic*, 2007.

[18] Masahiko Sato, Takafumi Sakurai, and Rod Burstall. Explicit environments. *Fundamenta Informaticae*, 45:1-2:79–115, 2001.

[19] Masahiko Sato, Takafumi Sakurai, Yukiyoshi Kameyama, and Atsushi Igarashi. Calculi of meta-variables. In *CSL*, volume 2803 of *LNCS*, pages 484–497, 2003.

[20] C. Urban, A. M. Pitts, and Murdoch J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.

[21] Christian Urban and Christine Tasson. Nominal techniques in Isabelle/HOL. In *CADE*, volume 3632 of *Lecture Notes in Artificial Intelligence*, pages 38–53, 2005.

[22] Johan van Benthem. Higher-order logic. In *Handbook of Philosophical Logic, 2nd Edition*, volume 1, pages 189–244. Kluwer, 2001.