ELSEVIER
CrossMark
ScienceDirect
Indagationes mathematicae
Indagationes Mathematicae 24 (2013) 1073–1104
www.elsevier.com/locate/indag

Viewing λ-terms through maps

Masahiko Sato[a], Randy Pollack[b,*], Helmut Schwichtenberg[c], Takafumi Sakurai[d]

[a] *Graduate School of Informatics, Kyoto University, Japan*
[b] *School of Engineering and Applied Sciences, Harvard University, United States*
[c] *Mathematics Institute, University of Munich, Germany*
[d] *Department of Mathematics and Informatics, Chiba University, Japan*

Abstract

In this paper we introduce the notion of *map*, which is a notation for the set of occurrences of a symbol in a syntactic expression such as a formula or a λ term. We use binary trees over 0 and 1 as maps, but some well-formedness conditions are required. We develop a representation of lambda terms using maps. The representation is concrete (inductively definable in HOL or Constructive Type Theory) and canonical (one representative per λ term). We define substitution for our map representation, and prove the representation is in substitution preserving isomorphism with both nominal logic λ terms and de Bruijn nameless terms. These proofs are mechanically checked in Isabelle/HOL and Minlog respectively.

The map representation has good properties. Substitution does not require adjustment of binding information: neither α conversion of the body being substituted into, nor de Bruijn lifting of the term being implanted. We have a natural proof of the substitution lemma of λ calculus that requires no fresh names, or index manipulation.

Using the notion of map we study conventional raw λ syntax. E.g. we give, and prove correct, a decision procedure for α equivalence of raw λ terms that does not require fresh names.

We conclude with a definition of β reduction for map terms, some discussion on the limitations of our current work, and suggestions for future work.

Keywords: Binding; de Bruijn; Nameless dummies; Lambda calculus; Nominal logic; Formal proof; Machine checked proof

* Corresponding author.
E-mail addresses: masahiko@kuis.kyoto-u.ac.jp (M. Sato), rpollack@seas.harvard.edu, rpollack@inf.ed.ac.uk (R. Pollack), schwicht@math.lmu.de (H. Schwichtenberg), sakurai@math.s.chiba-u.ac.jp (T. Sakurai).

0019-3577/$ - see front matter
http://dx.doi.org/10.1016/j.indag.2013.08.003

1. Introduction

In this paper we introduce the notion of *map* which is a generalization of the notion of *occurrence* of a symbol in syntactic expressions such as formulas or λ terms. We use binary trees over 0 and 1 as maps. For example, consider a λ term $(xz)(yz)$ in which each of the symbols x and y occurs once and the symbol z occurs twice; we use $(10)(00)$, $(00)(10)$ and $(01)(01)$ to represent the occurrences of the symbols x, y and z respectively. The bound positions are represented only by a constant $\Box$ (called *box*). We will write (omitting some parentheses for readability)

$$(10\ 00)\backslash(00\ 10)\backslash(01\ 01)\backslash(\Box\Box\ \Box\Box)$$

for the S combinator $\lambda xyz.\,(xz)(yz)$. $\Box$ may also occur unbound. Free variables may still occur in terms, e.g. the informal term $\lambda z.(xz)$ is written as $01\backslash(x\ \Box)$, but there are no bound names or de Bruijn indices.

Some well-formedness conditions will be required (Sections 2 and 3). Since we want a *canonical* representation (one notation per lambda term), although $\lambda x\,x.\,x$ is accepted as a correct notation for a lambda term, we will not accept $1\backslash 1\backslash\Box$ as a correct notation. Also, consider the substitution of $(\Box\ \Box)$ for the position 10 (the first $\Box$) in $01\backslash(\Box\ \Box)$; we get $01\backslash((\Box\ \Box)\ \Box)$ which does not match the intuition hinted at above because 0 is not a position in $(\Box\ \Box)$. For this reason we identify the map $(0\ 0)$ with the map 0, as discussed in Section 2.

1.1. Three abstraction mechanisms

In this paper we study three abstraction mechanisms and the three associated representations of lambda terms: Λ of *raw* λ *terms* [2], $\mathbb{L}$ of *map* λ*-expressions* and $\mathbb{D}$ of *de Bruijn-expressions* [5], generated by the following grammar:

$$\begin{array}{rl}
K, L \in \Lambda & ::= x \mid \Box \mid \mathsf{app}(K, L) \mid \mathsf{lam}(x, K) \\
M, N \in \mathbb{L} & ::= x \mid \Box \mid \mathsf{app}(M, N) \mid \mathsf{mask}(m, M) \quad (\text{where } m \mid M) \\
D, E \in \mathbb{D} & ::= x \mid \Box \mid \mathsf{app}(D, E) \mid i \mid \mathsf{bind}(D) \\
x \in \mathbb{X} & \text{The type of } \textit{atoms} \text{ or } \textit{parameters} \\
i \in \mathbb{I} & \text{The type of } \textit{natural numbers}, \text{ used as indices} \\
m \in \mathbb{M} & \text{The type of } \textit{maps}.
\end{array}$$

The three abstraction mechanisms of these three representations are:

Lambda abstraction This is abstraction by *parameters*, and is realized by the constructor lam (λ) in Λ. Quotienting by α equivalence is needed to make this mechanism work. The information about binding is shared between binding occurrences and bound occurrences (as shared names), and substitution may require adjusting both binding occurrences and bound occurrences of the base term (α-conversion).

Mask-abstraction This is abstraction by *maps*, and is realized by the constructor mask in $\mathbb{L}$. (We write $m\backslash M$ for $\mathsf{mask}(m, M)$.) For this representation to work, mask must be guarded by a simultaneously defined relation, written $m \mid M$, which is explained in Section 3. The information about binding is carried only at the binding occurrences (as maps). It will be seen that substitution does not require any adjustment of binding information.

Bind-abstraction This is abstraction by *indices*, and is realized by the constructor bind in $\mathbb{D}$. The information about binding is carried only at the bound occurrences (as indices pointing

to the binding point). Substitution requires adjustment of the implanted term (de Bruijn lifting).

The three types, Λ, $\mathbb{L}$ and $\mathbb{D}$ all have *parameters* $x \in \mathbb{X}$ and $\Box$ (*box*) as atomic objects, and have the constructor app in common. (See Remark 2 for further comment on the use of unbound $\Box$ in mask-abstraction.)

We will compare these three types through maps. Our strategy is to use $\mathbb{L}$ as the main type by which the types Λ and $\mathbb{D}$ are analyzed. The main results of the paper will show that Λ quotiented by the α equivalence relation, and the datatype $\mathbb{D}$, are both isomorphic to $\mathbb{L}$. Here, by an isomorphism we mean a bijection which respects constants, application and substitution.

1.2. Some properties of $\mathbb{L}$

The datatype $\mathbb{L}$ enjoys good properties. First, the closed expressions in $\mathbb{L}$ (possibly containing unbound $\Box$) are constructed from a finite set of elements, whereas the traditional approach requires an infinite set of variables even to represent all closed terms. For example, to construct the S combinator $\lambda xyz.\,(xz)(yz)$, one must first construct $(xz)(yz)$ containing three distinct *free* variables. In our approach $\mathsf{S} = (10\,00)\backslash(00\,10)\backslash(01\,01)\backslash(\Box\Box\;\Box\Box)$, can be constructed from the expression $(\Box\Box\;\Box\Box)$ by abstracting the three maps (01 01), (00 10), and (10 00). (Our language of maps is infinite.) Also note that we can compute the three maps from $\lambda xyz.\,(xz)(yz)$, but cannot recover the three variables x, y, z from our representation of the S combinator since, in the traditional approach, there are infinitely many α-equivalent representations of S. Of course de Bruijn nameless terms have these properties too.

A property that distinguishes $\mathbb{L}$ from $\mathbb{D}$ is the structural induction principle for $\mathbb{L}$ terms. de Bruijn nameless terms, as a concrete datatype, have a structural induction principle, but it does not capture the intended reading of the concrete structure as representing binding. A classic example of the use of term induction is the proof of the *substitution lemma*. For de Bruijn terms, this lemma is proved by structural induction, but with tricky lemmas about adjustment of indexes. Even worse, for named representations like nominal logic, a proof of the substitution lemma requires choosing a fresh name and α converting. In Lemma 2 we give a proof by structural induction over $\mathbb{L}$ terms and equational reasoning; no name or index adjustment is required.

1.3. Formal development

Much of the work in this paper is formalized and mechanically checked in Isabelle/HOL/Nominal [23] and/or in Minlog [21]. Section 4.5 outlines a formal development of the adequacy of the representation $\mathbb{L}$ with respect to lambda terms in Nominal Isabelle; i.e. an isomorphism that respects substitution. Naturally this proof had to be done using Isabelle. Section 5 describes a proof of adequacy of the representation $\mathbb{L}$ with respect to lambda terms in de Bruijn nameless notation. This proof is formalized in Minlog, due to expertise and preferences of the authors. The proof developments are available online from http://homepages.inf.ed.ac.uk/rpollack/export/SatoPollackSchwichtenbergSakurai-isabelle.tgz,SatoPollackSchwichtenbergSakurai-minlog.tgz.

2. The types $\mathbb{X}$, $\mathbb{I}$ and $\mathbb{M}$

In this section we introduce the datatype $\mathbb{M}$ of *maps* which will be used throughout the paper. We also mention the type $\mathbb{X}$ of *parameters* and the type of natural numbers, $\mathbb{I}$, used as *indices*.

$$\frac{}{\mathsf{one} \in \mathbb{M}^+} \qquad \frac{m^+ \in \mathbb{M}^+}{\mathsf{inl}(m^+) \in \mathbb{M}^+} \qquad \frac{n^+ \in \mathbb{M}^+}{\mathsf{inr}(n^+) \in \mathbb{M}^+} \qquad \frac{m^+ \in \mathbb{M}^+ \quad n^+ \in \mathbb{M}^+}{\mathsf{cons}(m^+, n^+) \in \mathbb{M}^+}$$

$$\frac{}{\mathsf{zero} \in \mathbb{M}}\ \mathsf{zero} \qquad \frac{m^+ \in \mathbb{M}^+}{m^+ \in \mathbb{M}}\ \mathit{incl}$$

Fig. 1. The datatype $\mathbb{M}$.

We use i, j, k as metavariables ranging over $\mathbb{I}$. We reserve and use the box symbol $\Box$ as a special constant denoting a hole to be filled with other expressions.

Fix a countably infinite set of atoms $\mathbb{X}$ for *global* (free) variables (also called *parameters*), and assume only that equality between parameters is decidable. We use x, y, z, as metavariables ranging over parameters. We adopt the polymorphic notation $x \sharp \Sigma$ from nominal logic.[1] In our map representation (as in well known representations such as locally nameless [1]) there is no "binding" in the nominal sense, and $x \sharp \Sigma$ means parameter x literally does not occur in structure Σ of whatever type.

2.1. *The datatype* $\mathbb{M}$ *of maps*

The type $\mathbb{M}$ of *maps* will be used to realize the abstraction mechanism in our target domain $\mathbb{L}$. One can obtain the domain $\mathbb{M}$ from the domain of Lisp symbolic expressions (binary trees) generated from the two atoms 0 and 1, by making the identification $\mathrm{cons}(0, 0) = 0$. This kind of symbolic expression with the property $\mathrm{cons}(0, 0) = 0$ was introduced by the first author of this paper in Sato–Hagiya [18] and Sato [16].

In order to formalize $\mathbb{M}$ as a datatype (Fig. 1) we introduce an auxiliary datatype $\mathbb{M}^+$ of non-zero maps. We can also represent $\mathbb{M}$ by the following unambiguous context-free grammar.

$$\begin{aligned} \mathbb{M} &::= \mathsf{zero} \mid m^+ \\ m^+, n^+ \in \mathbb{M}^+ &::= \mathsf{one} \mid \mathsf{inl}(m^+) \mid \mathsf{inr}(m^+) \mid \mathsf{cons}(m^+, n^+). \end{aligned}$$

Notational convention 1. We use m, n, p etc. as metavariables ranging over maps. We write $\mathsf{0}$ for zero and $\mathsf{1}$ for one. In the formal development we occasionally must pay attention to the difference between $\mathbb{M}$ and $\mathbb{M}^+$, but we suppress it in the rest of this paper.

As a general notational convention, throughout the paper we use sans-serif font for constructor functions and *slant* font for non-constructor functions. For example, in Fig. 1, the first five rules are all constructor rules which are used to construct new objects, while the last rule *incl* is a non-constructor rule used to include already constructed non-zero maps as elements in $\mathbb{M}$.

An important function $\mathit{mapp} : \mathbb{M} \times \mathbb{M} \to \mathbb{M}$ is defined as follows.

$$\mathit{mapp}(m, n) := \begin{cases} \mathsf{0} & \text{if } m = n = \mathsf{0}, \\ \mathsf{inl}(m) & \text{if } m \neq \mathsf{0} \text{ and } n = \mathsf{0}, \\ \mathsf{inr}(n) & \text{if } m = \mathsf{0} \text{ and } n \neq \mathsf{0}, \\ \mathsf{cons}(m, n) & \text{if } m \neq \mathsf{0} \text{ and } n \neq \mathsf{0}. \end{cases}$$

[1] In our nominal Isabelle formalization we use the same type of nominal atoms for $\mathbb{X}$ and for the names in nominal lambda terms; $\sharp$ is the Isabelle nominal freshness relation.

$$\frac{}{x \in \mathbb{L}} \qquad \frac{}{\Box \in \mathbb{L}} \qquad \frac{M \in \mathbb{L} \quad N \in \mathbb{L}}{\mathsf{app}(M,N) \in \mathbb{L}} \qquad \frac{m \in \mathbb{M} \quad M \in \mathbb{L} \quad m \mid M}{\mathsf{mask}(m,M) \in \mathbb{L}}$$

$$\frac{}{\mathsf{0} \mid x} \qquad \frac{}{\mathsf{0} \mid \Box} \qquad \frac{}{\mathsf{1} \mid \Box} \qquad \frac{m \mid M \quad n \mid N}{mapp(m,n) \mid \mathsf{app}(M,N)} \qquad \frac{m \mid N \quad n \mid N \quad m \perp n}{m \mid \mathsf{mask}(n,N)}$$

Fig. 2. Simultaneous inductive definition of the type $\mathbb{L}$ and the divisibility relation $\mid \subseteq \mathbb{M} \times \mathbb{L}$.

It is easily seen that *mapp* is injective. We write $(m\ n)$ and also mn for $mapp(m,n)$. Moreover, we write $(m\ n\ p)$ for $((m\ n)\ p)$ and mnp for $(mn)p$. For example, $(\mathsf{0}\ \mathsf{0}\ \mathsf{0}) = ((\mathsf{0}\ \mathsf{0})\ \mathsf{0}) = (\mathsf{0}\ \mathsf{0})\mathsf{0} = \mathsf{00} = \mathsf{0}$.

We define the *orthogonality* relation $\perp$ on $\mathbb{M}$ by the following inductive definition:

$$\frac{}{m \perp \mathsf{0}} \qquad \frac{}{\mathsf{0} \perp n} \qquad \frac{m \perp n \quad m' \perp n'}{mm' \perp nn'}$$

Note that $\perp$ is symmetric, and $\mathsf{0} \perp n$ for every map n. We can easily verify that if $\mathsf{1} \perp m$, then $m = \mathsf{0}$.

3. The datatype $\mathbb{L}$ of lambda expressions

Our target domain $\mathbb{L}$ of *map* λ*-expressions* (or, simply, *lambda expressions*) is defined by the rules in Fig. 2. In this definition, we use maps in the rule which constructs lambda expressions $\mathsf{mask}(m, M) \in \mathbb{L}$, called *abstracts*. In this rule, there is a third premise $m \mid M$ (read *m divides M*) which allows the construction of an abstract $m\backslash M$ from m and M only when this *divisibility condition* is satisfied.

We can also define $\mathbb{L}$ by the following grammar.

$$M, N \in \mathbb{L} ::= x \mid \Box \mid \mathsf{app}(M,N) \mid \mathsf{mask}(m,M) \quad (m \mid M).$$

This grammar is not context-free since $\mathsf{mask}(m, M)$ is accepted only if $m \mid M$. The grammar is however unambiguous and the syntactic objects defined by the grammar correspond bijectively to lambda expressions inductively defined in Fig. 2. Thus, the principle of 'what you see is what you get' applies to lambda expressions.

Remark 1. The rules of Fig. 2 do not fit the usual notion of "simultaneous inductive definition" since $\mathbb{L}$ occurs in the type of divisibility, although $\mathbb{L}$ is being defined simultaneously with divisibility. This kind of definition is called *inductive–inductive definition* (Forsberg and Setzer [7]). Since divisibility can also be viewed as a Boolean-valued function defined by recursion over $\mathbb{L}$, Fig. 2 could be reformulated as a *simultaneous inductive–recursive definition* (Dybjer [6]).

Further, since the relation $m \mid M$ is decidable, one can consider a formalization where the constructor mask only takes the first two arguments, and the third (proof) argument is irrelevant.

Since we do not use a formal proof tool that supports inductive–inductive or inductive–recursive definition, we give a conventional definition (Section 3.1) of $\mathbb{L}$ as a subset (predicate) of a datatype of *symbolic expressions*.

Notational convention 2. We use M, N, P as metavariables ranging over lambda expressions. We write $(M\ N)$ and also MN for $\mathsf{app}(M,N)$. We write $(m\backslash M)$ and also $m\backslash M$ for $\mathsf{mask}(m,M)$. A lambda expression of the form $\mathsf{mask}(m,M)$ is called an *abstract*. We use A, B as metavariables

$$\frac{}{x \in \mathbb{S}}\ par \qquad \frac{}{\Box \in \mathbb{S}}\ box \qquad \frac{S \in \mathbb{S} \quad T \in \mathbb{S}}{\mathsf{sapp}(S,T) \in \mathbb{S}}\ \mathsf{sapp} \qquad \frac{m \in \mathbb{M} \quad S \in \mathbb{S}}{\mathsf{smask}(m,S) \in \mathbb{S}}\ \mathsf{smask}$$

Fig. 3. Inductive definition of the datatype $\mathbb{S}$.

$$\frac{}{\mathsf{0} \mid x} \qquad \frac{}{\mathsf{0} \mid \Box} \qquad \frac{}{\mathsf{1} \mid \Box} \qquad \frac{m \mid S \quad n \mid T}{mn \mid ST} \qquad \frac{m \mid T \quad n \mid T \quad m \perp n}{m \mid n\backslash T}$$

$$\frac{}{x \in \mathsf{L}} \qquad \frac{}{\Box \in \mathsf{L}} \qquad \frac{M \in \mathsf{L} \quad N \in \mathsf{L}}{MN \in \mathsf{L}} \qquad \frac{m \in \mathbb{M} \quad M \in \mathsf{L} \quad m \mid M}{m\backslash M \in \mathsf{L}}$$

Fig. 4. Definitions of $\mathsf{ML} \subseteq \mathbb{M} \times \mathbb{S}$ (written $\cdot \mid \cdot$) and $\mathsf{L} \subseteq \mathbb{S}$.

ranging over abstracts, and write $\mathbb{A}$ for the subset of $\mathbb{L}$ consisting of all the abstracts.

3.1. The datatype $\mathbb{S}$ and embedding of $\mathbb{L}$ in $\mathbb{S}$

In the definition of $\mathbb{L}$ above, we used an auxiliary divisibility relation $|$, defining $\mathbb{L}$ and $|$ by simultaneous inductive–inductive definition. This approach is foundationally nice, since no extra objects are involved in the construction of $\mathbb{L}$ from the types $\mathbb{M}$ and $\mathbb{X}$. However, in order to formalize our ideas in a mechanical proof system, we now take another route to define the domain $\mathbb{L}$. In this approach, we first define a datatype $\mathbb{S}$ of *symbolic-expressions*, as shown in Fig. 3. We will then realize $\mathbb{L}$ within $\mathbb{S}$ by defining a subset L of $\mathbb{S}$ which is isomorphic to $\mathbb{L}$.

Unlike $\mathsf{mask} : \mathbb{M} \times \mathbb{L} \to \mathbb{L}$ (Fig. 2) which is partial, smask is a total binary operation on maps and symbolic-expressions. Every symbolic-expression is uniquely generated from $\Box$, $\mathbb{X}$ and $\mathbb{M}$ by finitely many applications of sapp and smask.

Notational convention 3. We use S, T as metavariables ranging over $\mathbb{S}$. We write $(S\ T)$ and also ST for $\mathsf{sapp}(S, T)$. We write $(m\backslash S)$ and also $m\backslash S$ for $\mathsf{smask}(m, S)$.

Now we inductively define a relation $\mathsf{ML} \subseteq \mathbb{M} \times \mathbb{S}$, written $\cdot \mid \cdot$, and then a subset L of "well-formed" elements of $\mathbb{S}$, as shown in Fig. 4. We call the elements of L *symbolic lambda expressions*, and use M, N, P as metavariables ranging over symbolic lambda expressions. Relations L and ML are not simultaneously defined; ML is completely defined on its own, and carries all the information that is interesting:

- $m \mid S$ means "S is well-formed and m is a position of unbound boxes in S",
- $\mathsf{0} \mid S$ means "S is well-formed".

We show the definition of L in Fig. 4 in order to point out the relationship with the definition of $\mathbb{L}$.

Definition 1 (*Abstracts in* L)**.** A symbolic lambda expression is called an *abstract* if it is of the form $m\backslash M$. (Recall this implies $m \mid M$.) Metavariables A, B, C range over abstracts, and we write A for the set of abstracts in L.

Definition 2 (*Hole Filling and Instantiation*)**.** We write $M_m[P]$ for $\mathit{fill}(M, m, P)$, the result of *filling* the boxes (holes) in M specified by map m with P. $M_m[P]$ is defined only if $m \mid M$. We

write $A \triangledown P$ for the result of *instantiating* abstract A with P.

$$
\begin{array}{rcll}
\mathit{fill} & : & \mathsf{L} \times \mathbb{M} \times \mathsf{L} \to \mathsf{L} & \\
\Box_1[P] & := & P & \\
\Box_0[P] & := & \Box & \\
x_0[P] & := & x & \\
(M\ N)_{(m\ n)}[P] & := & (M_m[P]\ N_n[P]) & \text{if } m \mid M \text{ and } n \mid N \\
(n\backslash N)_m[P] & := & n\backslash(N_m[P]) & \text{if } m \mid (n\backslash N) \\
 & & & \\
\triangledown & : & \mathsf{A} \times \mathsf{L} \to \mathsf{L} & \\
(m\backslash M)\triangledown P & := & M_m[P]. &
\end{array}
$$

To see why the last equation defining *fill* is well formed and correct, note that $m \perp n$, so n has 0 in every position bound by m (where copies of P will be implanted). Thus $n \mid N_m[P]$, and n does not capture any positions in implanted copies of P.

Hole filling respects the definition of L:

$$m \mid M \wedge 0 \mid N \Longrightarrow 0 \mid M_m[N].$$

Hole filling is a homomorphism, going under a binder without the need to adjust the abstractor (as needed with nominal logic and raw λ syntax) or the object being implanted (as needed with de Bruijn index representation).

3.2. The use of parameters in L

Parameters are necessary to express open terms. In conventional presentations of binding, parameters can become bound, and can be substituted for [1,2,4,13,14,23]. These cited works differ in degree of formality, and in the mechanism of binding and substitution, but all use names in some way.

We define functions *map* and *skeleton* in Definition 3, showing how parameters can be used in our representation L. $\mathit{map} : \mathbb{X} \times \mathsf{L} \to \mathbb{M}$ computes the map of all the occurrences of a parameter in a symbolic lambda term; $\mathit{skel} : \mathbb{X} \times \mathsf{L} \to \mathsf{L}$ replaces all occurrences of a parameter in a symbolic lambda term with $\Box$. Together *map* and *skel* are used to abstract a parameter from a symbolic lambda term, and to define substitution for a parameter (Definition 4). The use of *map* and *skel* to represent abstracts goes back to [18,16,17], but the map part is greatly simplified in this paper.

Definition 3 (*Map and Skeleton*)**.** We write M_x for $\mathit{map}(x, M)$, and M^x for $\mathit{skel}(x, M)$.

$$
\begin{array}{rcl@{\qquad}rcl}
\mathit{map} & : & \mathbb{X} \times \mathsf{L} \to \mathbb{M} & \mathit{skel} & : & \mathbb{X} \times \mathsf{L} \to \mathsf{L} \\
y_x & := & \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y. \end{cases} & y^x & := & \begin{cases} \Box & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases} \\
\Box_x & := & 0 & \Box^x & := & \Box \\
(M\ N)_x & := & (M_x\ N_x) & (M\ N)^x & := & (M^x\ N^x) \\
(m\backslash M)_x & := & M_x & (m\backslash M)^x & := & m\backslash M^x.
\end{array}
$$

It is easy to see that (1) $M_x \mid M^x$ and that (2) $M_x = 0$ if and only if $M^x = M$. (See also Proposition 4.)

Definition 4 (*Lambda Abstraction and Substitution*)**.**

$$lam \;:\; \mathbb{X} \times \mathsf{L} \to \mathsf{A} \qquad subst \;:\; \mathsf{L} \times \mathbb{X} \times \mathsf{L} \to \mathsf{L}$$
$$lam(x, M) \;:=\; M_x \backslash M^x \qquad M\{x \backslash N\} \;:=\; lam(x, M) \triangledown N.$$

Unlike mask (Figs. 2 and 4), *lam* is not injective (e.g. $lam(x, x) = lam(y, y)$), but we have:

$$lam(x, M) = lam(x, N) \Longrightarrow M = N.$$

Remark 2. Now we can draw a comparison between the mask abstraction of our representation, and the abstraction of various local representations [1,11,14] that may illuminate the rôle of unbound boxes. For example consider the locally nameless representation [1]. The two constructors of well-formed lambda terms from smaller well-formed terms are application and abstraction. Application just puts two well-formed terms next to each other, and this construction is shared by the two representations.

More interestingly, abstraction takes a well-formed lambda term and some information about what is to be abstracted from it. In mask abstraction, this "information about what is to be abstracted" is exactly a map, and abstraction is just putting a map and a well-formed lambda term next to each other. This requires that "well-formed" allows the presence of unbound boxes. However in well-formed locally nameless terms, unbound indices are not allowed (this would negate the point of locally nameless terms, namely that they are "index closed" so do not require de Bruijn lifting for manipulations such as substitution). Abstraction of a locally nameless term is not placing an abstractor, λ, next to a well formed term, t, but requires replacing a free name in t (that is to be bound) by the appropriate index to be bound by an outermost λ.

We could, if desired, remove unbound boxes from the notion of well-formed mask terms, using a function such as *lam* of Definition 4 that replaces a free name by boxes, similarly to the abstraction of [1,11,14], but we would lose the name-free aspect of mask terms, and they would behave very much like the local representations [1,11,14].

Conversely, could we modify a local representation such as [1,11,14] to be name-free by allowing some other constants (analogous to unbound $\Box$) in well-formed terms? The answer seems to be no, as all the cited local representations depend on having no free local names.

"Abstraction" is not a symmetrical operation: the body of the abstraction should be a well-formed term, but the abstractor is some other kind of information. Our bind abstraction is the only approach we know that uses this "other kind of information" in the abstractor to completely specify what is to be abstracted, rather than sharing it between the body and the abstractor (e.g. nominal representation) or putting it entirely in the body (e.g. de Bruijn representation). That is why we can define abstraction without having to modify the body.

We can prove the following equations about substitution.

Proposition 1.

$$y\{x \backslash P\} = \begin{cases} P & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases}$$
$$\Box\{x \backslash P\} = \Box$$
$$(M\ N)\{x \backslash P\} = (M\{x \backslash P\}\ N\{x \backslash P\})$$
$$(m \backslash M)\{x \backslash P\} = m \backslash (M\{x \backslash P\}) \qquad \text{if } m \mid M$$

$$lam(y, M)\{x \backslash P\} = lam(y, M) \qquad \text{if } x = y \text{ or } x \sharp M$$
$$lam(y, M)\{x \backslash P\} = lam(y, M\{x \backslash P\}) \qquad \text{if } x \neq y \text{ and } y \sharp P$$
$$lam(y, M) = lam(z, M\{x \backslash z\}) \qquad \text{if } z \sharp M.$$

$$\frac{}{x \in \Lambda}\ par \qquad \frac{}{\Box \in \Lambda}\ box \qquad \frac{K \in \Lambda \quad L \in \Lambda}{\mathsf{app}(K, L) \in \Lambda}\ \mathsf{app} \qquad \frac{K \in \Lambda}{\mathsf{lam}(x, K) \in \Lambda}\ \mathsf{lam}$$

Fig. 5. Definition of the datatype Λ.

To see why the fourth equation is well formed, note that since $m \mid M$, if x occurs in M then the corresponding positions in m must be $\mathtt{0}$. Thus the right hand side is well formed and does not capture any positions in P.

The first four equations of Proposition 1 show how to completely evaluate any L term containing a substitution. Unlike the last three equations (which are often used in the standard definition of substitution on λ terms), the first four rules have no freshness side condition. Thus, in our system, substitutions can be eliminated without α-conversion.[2] As an example we mention the usual substitution lemma.

Lemma 2 (*Substitution Lemma*)**.** *If $x \neq y$ and $x \sharp P$, then*

$$M\{x\backslash N\}\{y\backslash P\} = M\{y\backslash P\}\{x\backslash N\{y\backslash P\}\}.$$

Proof. By induction on $M \in \mathbb{L}$, using Proposition 1 to compute the substitution operation without choosing fresh names. The two side conditions of the lemma are used only in the parameter case; the interesting case of abstraction is proved by equational reasoning using the induction hypothesis. □

Compare this proof with that given in Urban [23] for informal α-equated lambda terms: when $M = \lambda z.M'$ we must assume $z \sharp (x, y, N, P)$. In the same paper Urban shows a formal proof of the proposition for nominal lambda terms that is automated and slick, but still proceeds by choosing a sufficiently fresh variable in the abstraction case.

We conclude this section by giving a lemma which will be used in the proof of Theorem 16.

Lemma 3. *Suppose $z \neq x$, $P_z = \mathtt{0}$, $M_z = N_z$ and $M^z\{x\backslash P\} = N^z$. Then $M\{x\backslash P\} = N$.*

Proof. By induction on M. □

4. The datatype Λ of raw λ terms

We define the set Λ of *raw λ terms* as a datatype constructed by the rules in Fig. 5. We can also define Λ as the language characterized by the following context-free grammar.

$$K, L \in \Lambda ::= x \mid \Box \mid \mathsf{app}(K, L) \mid \mathsf{lam}(x, K)$$
$$x \in \mathbb{X}.$$

Notational convention 4. We use J, K, L as metavariables ranging over raw λ terms. We write $(K\ L)$ and also $K\,L$ for $\mathsf{app}(K, L)$.

[2] Other representations have this same property [11,1,14].

As observed by Pitts and Gabbay [8,13], the notion of *equivariance* plays a key role in studying the datatype Λ. We quickly review the notion here. Let $G_{\mathbb{X}}$ be the group of finite permutations on $\mathbb{X}$. Suppose that $G_{\mathbb{X}}$ acts on two sets U and V and let $f : U \to V$. The map f is said to be an *equivariant map* if f commutes with all $\pi \in G_{\mathbb{X}}$ and $u \in U$, namely, $f(u^\pi) = f(u)^\pi$ where we write $(-)^\pi$ for the action of π on $(-)$. We will also write $x /\!\!/ y$ for the permutation which (only) swaps x and y. We can naturally define the group action of $G_{\mathbb{X}}$ on all the objects we introduce in this paper including, in particular, objects in Λ, $\mathbb{L}$ and $\mathbb{D}$. Then all the functions and relations we introduce have the equivariance property. Formally this must be proved for particular functions and relations for which it is required, but the essential reason why this holds is that we define functions etc. without mentioning any particular atom, relying only on the fact that the equality relation on $\mathbb{X}$ is decidable and that $\mathbb{X}$ contains infinitely many parameters. For example, the constructors app and lam for the datatype Λ are equivariant maps.

Definition 5 (*Free Parameters*)**.** We define the set $\mathrm{FP}(K)$ of *free parameters in* K as follows.

$$\begin{aligned} \mathrm{FP}(x) &:= \{x\} \\ \mathrm{FP}(\Box) &:= \{\} \\ \mathrm{FP}(K\,L) &:= \mathrm{FP}(K) \cup \mathrm{FP}(L) \\ \mathrm{FP}(\mathsf{lam}(x, K)) &:= \mathrm{FP}(K) \setminus \{x\}. \end{aligned}$$

We mean Λ to be pure (inductively defined) syntax. Thus $x \sharp K$ implies $x \notin \mathrm{FP}(K)$ but the converse is not true in general.

4.1. Map/skeleton functions on Λ

Definition 6 (*Map and Skeleton*)**.** We define two functions *map* and *skel*, reminiscent of Definition 3. We write K_x for $map(x, K)$, and K^x for $skel(x, K)$.

$$\begin{array}{rcl} map & : & \mathbb{X} \times \Lambda \to \mathbb{M} \\ y_x & := & \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{if } x \neq y \end{cases} \\ \Box_x & := & 0 \\ (K\ L)_x & := & (K_x\ L_x) \\ \mathsf{lam}(y, K)_x & := & \begin{cases} 0 & \text{if } x = y, \\ K_x & \text{if } x \neq y \end{cases} \end{array} \qquad \begin{array}{rcl} skel & : & \mathbb{X} \times \Lambda \to \Lambda \\ y^x & := & \begin{cases} \Box & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases} \\ \Box^x & := & \Box \\ (K\ L)^x & := & (K^x\ L^x) \\ \mathsf{lam}(y, K)^x & := & \begin{cases} \mathsf{lam}(y, K) & \text{if } x = y, \\ \mathsf{lam}(y, K^x) & \text{if } x \neq y. \end{cases} \end{array}$$

We can characterize the relation $x \in \mathrm{FP}(K)$ using map and skeleton.

Proposition 4 (*Freshness*)**.** $x \notin \mathrm{FP}(K) \iff K_x = 0 \iff K^x = K$.

Lemma 5 (*Simple Properties of map and skel*)**.**

1. $(K^x)^y = (K^y)^x$.
2. *If* $x \neq y$ *then* $(K^x)_y = K_y$.
3. $(K^x)^x = K^x$ *and* $(K^x)_x = 0$.

4.2. α equivalence

Definition 7 (α *Equivalence*)**.** We define the α equivalence relation, $=_\alpha$, using the map/skeleton functions (Definition 6) as shown in Fig. 6.

$$\frac{}{x =_\alpha x} \qquad \frac{}{\square =_\alpha \square} \qquad \frac{K =_\alpha K' \quad L =_\alpha L'}{(K\ L) =_\alpha (K'\ L')} \qquad \frac{K_x = L_y \quad K^x =_\alpha L^y}{\mathsf{lam}(x, K) =_\alpha \mathsf{lam}(y, L)}$$

Fig. 6. Definition of the α equivalence relation.

It is easy to see that $=_\alpha$ is a decidable equivalence relation. It is interesting that we can decide α equivalence of raw syntax without any renaming. For example, we can show that $\mathsf{lam}(x, \mathsf{lam}(y, yx)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(x, xy))$ as follows.

$$\dfrac{01 = 01 \qquad \dfrac{10 = 10 \qquad \dfrac{\dfrac{}{\square =_\alpha \square} \quad \dfrac{}{\square =_\alpha \square}}{\square\square =_\alpha \square\square}}{\mathsf{lam}(y, y\square) =_\alpha \mathsf{lam}(x, x\square)}}{\mathsf{lam}(x, \mathsf{lam}(y, yx)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(x, xy))}$$

Of course a similar end can be accomplished by translation to de Bruijn nameless representation, but we can do this staying in raw lambda terms (with the special constant $\square$).

The following lemma establishes the congruence of $=_\alpha$, i.e. the constructors app and lam are well-defined on $=_\alpha$ equivalence classes.

Lemma 6. *Suppose $K =_\alpha L$. Then:*
1. *$K^z =_\alpha L^z$ and $K_z = L_z$,*
2. *$\mathsf{lam}(z, K) =_\alpha \mathsf{lam}(z, L)$.*

Proof. Show 1 by induction on the derivation of $K =_\alpha L$; 2 follows from 1. $\square$

In the rest of this subsection we outline a proof that $=_\alpha$ is equivalent to a standard definition of α equivalence. The reader may be interested in papers giving other relations deciding α equivalence without renaming [22,10,12,20,9]. The discussion of α equivalence in Section 2 of [9] is especially interesting.

The following relation $\sim$ is introduced in Gabbay and Pitts [8]:

$$\frac{}{x \sim x} \qquad \frac{}{\square \sim \square} \qquad \frac{K \sim K' \quad L \sim L'}{(K\ L) \sim (K'\ L')} \qquad \frac{K^{x\#z} \sim L^{y\#z} \quad z \sharp \{x, y, K, L\}}{\mathsf{lam}(x, K) \sim \mathsf{lam}(y, L)}$$

Proposition 2.2 in [8] proves that $\sim$ coincides with a standard definition of α equivalence; here we show that $=_\alpha$ coincides with $\sim$. Our proof is formalized in Isabelle/HOL. Although the algorithm of Fig. 6 does not use fresh variables, our proof of correctness does. The tricky proofs of Lemmas 8 and 10 are detailed in an Appendix.

Lemma 7. 1. *If $z \sharp K$ and $y \notin \{x, z\}$ then $(K^{x\#z})^y = (K^y)^{x\#z}$.*
2. *If $z \sharp K$ and $y \notin \{x, z\}$ then $(K^{x\#z})_y = (K^x)_y$.*
3. *If $z \sharp K$ then $(K^{x\#z})^z = (K^x)^{x\#z}$ and $(K^{x\#z})_z = K_x$.*
4. *If $z \sharp K$ then $(K^{x\#z})^x = K^{x\#z}$ and $(K^{x\#z})_x = 0$.*

Lemma 8. *If $z \sharp \{x, y, K, L\}$ then $K^{x\#z} =_\alpha L^{y\#z} \Longrightarrow K^x =_\alpha L^y$ and $K_x = L_y$.*

Proposition 9. *$K \sim L \Longrightarrow K =_\alpha L$.*

Lemma 10. *If $z \sharp \{x, y, K, L\}$ then $K^x \sim L^y$ and $K_x = L_y \Longrightarrow K^{x\#z} \sim L^{y\#z}$.*

$$\frac{}{x\{x\backslash J\} \to J} \qquad \frac{x \neq y}{y\{x\backslash J\} \to y} \qquad \frac{}{\Box\{x\backslash J\} \to \Box}$$

$$\frac{K\{x\backslash J\} \to L \quad K'\{x\backslash J\} \to L'}{(K\ K')\{x\backslash J\} \to (L\ L')} \qquad \frac{z \notin \{x\} \cup \mathrm{FP}(J) \quad K\{x\backslash J\} \to L}{\mathsf{lam}(z, K)\{x\backslash J\} \to \mathsf{lam}(z, L)}$$

$$\frac{K =_\alpha K' \quad J =_\alpha J' \quad K'\{x\backslash J'\} \to L' \quad L' =_\alpha L}{K\{x\backslash J\} \to L}$$

Fig. 7. Definition of the substitution relation.

Proposition 11. $K =_\alpha L \Longrightarrow K \sim L$.

4.3. Substitution

It is well-known that a choice function on names is required to define the substitution operation canonically on raw λ terms, due to the possibility of parameter capture. Here, we define substitution up to α equivalence, not as an operation but as a 4-ary relation.

Definition 8 (*Substitution*)**.** We define $\mathit{Subst} \subseteq \Lambda \times \mathbb{X} \times \Lambda \times \Lambda$ as shown in Fig. 7. We write $K\{x\backslash J\} \to L$ for $(J, x, K, L) \in \mathit{Subst}$.

We wish to show that the substitution relation enjoys the expected properties (Proposition 14 and Theorem 16). In the proofs below we will sometimes induct on the size of a derivation $\mathcal{D}$ asserting that a substitution relation holds, and we will write $|\mathcal{D}|$ for the size of the derivation. We will also use induction on the size of a raw lambda term K (a lambda expression M), and we will write $|K|$ ($|M|$) for the size of K (M, respectively).

We first prepare the following two lemmas.

Lemma 12. $y \notin \mathrm{FP}(K) \Longrightarrow \exists L.\ \mathsf{lam}(x, K) =_\alpha \mathsf{lam}(y, L)$ *and* $|K| = |L|$.

Proof. See the Appendix. □

Lemma 13. *If* $z \notin \{x\} \cup \mathrm{FP}(J)$ *and* $\mathcal{D}$ *proves* $K\{x\backslash J\} \to L$*, then* (1) $K_z = L_z$ *and* (2) *we can construct a derivation* $\mathcal{D}'$ *such that* $|\mathcal{D}'| = |\mathcal{D}|$ *and* $\mathcal{D}'$ *proves* $K^z\{x\backslash J\} \to L^z$.

Proof. By induction on $\mathcal{D}$ using Proposition 4 in the base case. □

Proposition 14 (*Properties of Substitution*)**.**

1. *(Existence)* $\exists L.\ K\{x\backslash J\} \to L$.
2. *(Uniqueness)*
$$(K\{x\backslash J\} \to L \wedge J =_\alpha J' \wedge K =_\alpha K' \wedge K'\{x\backslash J'\} \to L') \Longrightarrow L =_\alpha L'.$$
3. *(Congruence)*
$$(K\{x\backslash J\} \to L \wedge J =_\alpha J' \wedge K =_\alpha K' \wedge L =_\alpha L') \Longrightarrow K'\{x\backslash J'\} \to L'.$$

Proof. In this proof, we call the last rule of Fig. 7 the α-cut-rule.

Proof of 1. By induction on $|K|$. The crucial case is for lam: given $\mathsf{lam}(z, K)$, the goal is to find L such that $\mathsf{lam}(z, K)\{x\backslash J\} \to L$.

We can take z' such that $z' \notin \mathrm{FP}(K)$ and $z' \notin \{x\} \cup \mathrm{FP}(J)$. By Lemma 12, there exists K' such that $\mathsf{lam}(z, K) =_\alpha \mathsf{lam}(z', K')$ and $|K| = |K'|$. By IH, there exists L' such that $K'\{x\backslash J\} \to L'$. Therefore, we have $\mathsf{lam}(z', K')\{x\backslash J\} \to \mathsf{lam}(z', L')$. By α-cut-rule, we have $\mathsf{lam}(z, K)\{x\backslash J\} \to \mathsf{lam}(z', L')$.

Proof of 2. We prove this case by inspecting two derivations: $\mathcal{D}$ which proves the judgment $K\{x\backslash J\} \to L$ and $\mathcal{D}'$ which proves $K'\{x\backslash J'\} \to L'$. Use double induction on $|\mathcal{D}|$ and $|\mathcal{D}'|$.

We classify the cases by the last rules of the derivations $\mathcal{D}$ and $\mathcal{D}'$.

(1) One or both of the last rules are α-cut-rules: This case is reduced to other cases by considering the premise of the α-cut-rule.

(2) Both of the rules are lam-rules: Let

$$\frac{z \notin \{x\} \cup \mathrm{FP}(J) \quad K\{x\backslash J\} \to L}{\mathsf{lam}(z, K)\{x\backslash J\} \to \mathsf{lam}(z, L)} \qquad \frac{z' \notin \{x\} \cup \mathrm{FP}(J') \quad K'\{x\backslash J'\} \to L'}{\mathsf{lam}(z', K')\{x\backslash J'\} \to \mathsf{lam}(z', L')}$$

be the last rules of each derivation. Since $\mathsf{lam}(z, K) =_\alpha \mathsf{lam}(z', K')$, we have $K^z =_\alpha K'^{z'}$ and $K_z = K'_{z'}$. By the premise of each rule and Lemma 13, we have $K^z\{x\backslash J\} \to L^z$, $K_z = L_z$, $K'^{z'}\{x\backslash J'\} \to L'^{z'}$, and $K'_{z'} = L'_{z'}$. Since $K^z =_\alpha K'^{z'}$, we have $L^z =_\alpha L'^{z'}$ by IH. We also have $L_z = L'_{z'}$ from $K_z = K'_{z'}$, $K_z = L_z$, and $K'_{z'} = L'_{z'}$. Therefore, we have $\mathsf{lam}(z, L) =_\alpha \mathsf{lam}(z', L')$.

(3) Other cases: Easy.

Proof of 3. Clear from α-cut-rule. □

4.4. Interpretation of raw λ terms in L

Definition 9 (*Denotation*)**.** We define a function $[\![\cdot]\!] : \Lambda \to \mathsf{L}$ as follows. (Recall *lam* from Definition 4.)

$$\begin{aligned} [\![x]\!] &:= x \\ [\![\Box]\!] &:= \Box \\ [\![(K\ L)]\!] &:= ([\![K]\!]\ [\![L]\!]) \\ [\![\mathsf{lam}(x, K)]\!] &:= \mathit{lam}(x, [\![K]\!]). \end{aligned}$$

We say that $[\![K]\!]$ is the *denotation* of K in L; i.e. a raw λ term K is a name denoting *the* λ term $[\![K]\!]$. Thus, our view is that elements of L correspond bijectively to ideal λ terms, while, for example, $\mathsf{lam}(x, x)$ and $\mathsf{lam}(y, y)$ are two different names of the same λ term, $\mathsf{1}\backslash\Box$.

Theorem 16 below shows that L adequately represents the structure of Λ modulo α equivalence. In particular, the third claim of the theorem shows that the substitution relation in Λ is represented by the substitution operation in L. We first prepare the following lemma.

Lemma 15 (*Preservation of Map and Commutation of Skel with* $[\![-]\!]$)**.**

1. $[\![K]\!]_x = K_x$.
2. $[\![K]\!]^x = [\![K^x]\!]$.

Proof. By induction on K. □

Theorem 16 (*Properties of Denotation*)**.**

1. $M \in \mathsf{L} \Longrightarrow \exists K \in \Lambda.\ [\![K]\!] = M$.
2. $K =_\alpha L \iff [\![K]\!] = [\![L]\!]$.
3. $K\{x\backslash J\} \to L \iff [\![K]\!]\{x\backslash [\![J]\!]\} = [\![L]\!]$.

Proof. *Proof of* 1. By induction on $|M|$. Consider the case where $M = n\backslash N$. Choose a z such that $z \notin \mathrm{FP}(N)$ and fill N with z at n. Since $N_n[z]$ is of the same size as N, we have IH for it, namely, we have K such that $[\![K]\!] = N_n[z]$. Then we have $[\![\mathsf{lam}(z, K)]\!] = [\![K]\!]_z \setminus [\![K]\!]^z = n\backslash N = M$.

Proof of 2. By induction on $|K|$. We use IH and Lemma 15 in the case where $K = \mathsf{lam}(x, K')$ and $L = \mathsf{lam}(y, L')$ for both directions.

Proof of 3. ($\Longrightarrow$) By induction on $|\mathcal{D}|$ where $\mathcal{D}$ proves $K\{x\backslash J\} \to L$. We consider the case where $\mathcal{D}$ is of the form:

$$\frac{z \notin \{x\} \cup \mathrm{FP}(J) \quad \mathcal{D}'}{\mathsf{lam}(z, K)\{x\backslash J\} \to \mathsf{lam}(z, L)}$$

and $\mathcal{D}'$ proves $K\{x\backslash J\} \to L$. Then, by Lemma 13, we have (1) $K_z = L_z$ and (2) $\mathcal{D}''$ such that $|\mathcal{D}''| = |\mathcal{D}'|$ and $\mathcal{D}''$ proves $K^z\{x\backslash J\} \to L^z$. Since $|\mathcal{D}''| = |\mathcal{D}'| < |\mathcal{D}|$, we have IH for $\mathcal{D}''$, namely, $[\![K^z]\!]\{x\backslash [\![J]\!]\} = [\![L^z]\!]$. Now, our goal in this case is: $[\![\mathsf{lam}(z, K)]\!]\{x\backslash [\![J]\!]\} = [\![\mathsf{lam}(z, L)]\!]$. We can achieve the goal as follows.

$$\begin{aligned}
[\![\mathsf{lam}(z, K)]\!]\{x\backslash [\![J]\!]\} &= ([\![K]\!]_z \setminus [\![K]\!]^z)\{x\backslash [\![J]\!]\} && \text{by Definition 9}\\
&= K_z\backslash([\![K^z]\!]\{x\backslash [\![J]\!]\}) && \text{by Proposition 1 and Lemma 15}\\
&= L_z\backslash [\![L^z]\!] && \text{by (1) and IH}\\
&= [\![L]\!]_z \setminus [\![L]\!]^z && \text{by Lemma 15}\\
&= [\![\mathsf{lam}(z, L)]\!] && \text{by Definition 9.}
\end{aligned}$$

($\Longleftarrow$) By induction on $|K|$. The interesting case is where $K = \mathsf{lam}(y_1, K_1)$. In this case L must be of the form $\mathsf{lam}(z_1, L_1)$, and we have

$$[\![\mathsf{lam}(y_1, K_1)]\!]\{x\backslash [\![J]\!]\} = [\![\mathsf{lam}(z_1, L_1)]\!]$$

by assumption. Choose a parameter z such that $z \notin \{x\} \cup \mathrm{FP}(J K_1 L_1)$. Then, by Lemma 12, we can find K_2 and L_2 such that $|K_1| = |K_2|$,

$$(1)\ \mathsf{lam}(y_1, K_1) =_\alpha \mathsf{lam}(z, K_2) \quad \text{and} \quad (2)\ \mathsf{lam}(z_1, L_1) =_\alpha \mathsf{lam}(z, L_2).$$

Then, using (1) and (2), the assumption can be rewritten to

$$[\![\mathsf{lam}(z, K_2)]\!]\{x\backslash [\![J]\!]\} = [\![\mathsf{lam}(z, L_2)]\!].$$

By simplifying this, we have

$$(K_2)_z\backslash([\![(K_2)^z]\!]\{x\backslash [\![J]\!]\}) = (L_2)_z\backslash [\![(L_2)^z]\!].$$

Hence, $[\![(K_2)^z]\!]\{x\backslash [\![J]\!]\} = [\![(L_2)^z]\!]$. From this, by Lemma 3, we have

$$[\![K_2]\!]\{x\backslash [\![J]\!]\} = [\![L_2]\!].$$

Then, since $|K_2| < |\mathsf{lam}(z, K_2)| = |\mathsf{lam}(y_1, K_1)|$, by IH, we have a derivation $\mathcal{D}$ which proves $K_2\{x\backslash J\} \to L_2$. Using $\mathcal{D}$, we can construct the following derivation which achieves our goal in this case.

$$\frac{\mathsf{lam}(x_1, K_1) =_\alpha \mathsf{lam}(z, K_2) \quad J =_\alpha J \quad \mathcal{D}_1 \quad \mathsf{lam}(z, L_2) =_\alpha \mathsf{lam}(y_1, L_1)}{\mathsf{lam}(x_1, K_1)\{x\backslash J\} \to \mathsf{lam}(y_1, L_1)},$$

where $\mathcal{D}_1$ is:

$$\frac{z \notin \{x\} \cup \mathrm{FP}(J) \quad \mathcal{D}}{\mathsf{lam}(z, K_2)\{x\backslash J\} \to \mathsf{lam}(z, L_2).} \quad \square$$

It is to be noted that we proved Theorem 16 without using Proposition 14, and that we can use this theorem to give an alternative proof of Proposition 14. In fact, it is easy to see that Proposition 14 follows from Theorem 16.

4.5. *Formalized correctness of the representation* L *w.r.t. nominal Isabelle*

Theorem 16 can be read as the correctness of L if you believe that $\Lambda, =_\alpha$ and *Subst* are a correct representation of raw λ syntax and α equivalence. Now we outline a direct proof that L is in substitution preserving isomorphism with the nominal representation of λ terms in Urban's nominal package for Isabelle/HOL [23]. We have formalized this proof in Isabelle/HOL.

Consider a proof of the first claim in Theorem 16. We define an inverse of *denote*:

$$\begin{aligned} \lfloor\cdot\rfloor &: \mathsf{L} \to \Lambda \\ \lfloor x\rfloor &:= x \\ \lfloor\Box\rfloor &:= \Box \\ \lfloor(M_1\ M_2)\rfloor &:= (\lfloor M_1\rfloor\ \lfloor M_2\rfloor) \\ \lfloor m\backslash M\rfloor &:= \mathsf{lam}(x, \lfloor M_m[x]\rfloor) \quad \text{if } x \sharp M. \end{aligned}$$

In the last rule, to get a name for $m\backslash M$, we fill hole m in M with a fresh parameter x, recursively compute a name for that term (which is smaller than $m\backslash M$), then abstract x in the raw language. These equations define a function only if there is a *canonical* way to choose $x \sharp M$ (which is slightly more than we originally assumed about $\mathbb{X}$). Even if we satisfy that requirement (e.g. take $\mathbb{X}$ to be totally ordered), $\lfloor\cdot\rfloor$ is only a right inverse of $[\![\cdot]\!]$, since $[\![\cdot]\!]$ is not injective due to α-variance in Λ.

However, if we switch our view from raw lambda terms to nominal lambda terms, then we can prove that $\lfloor\cdot\rfloor$ is a well defined function with the given equations, that it is a two-sided inverse to $[\![\cdot]\!]$, and that this bijection preserves substitution (recall Definitions 2 and 4):

$$[\![K\{x\backslash J\}]\!] = \mathit{fill}([\![K]\!]_x, [\![K]\!]^x, [\![J]\!]) = \mathit{lam}(x, [\![K]\!])\triangledown[\![J]\!] = [\![K]\!]\{x\backslash[\![J]\!]\}. \tag{1}$$

We prove Eq. (1) by induction on K. In the abstraction case where $K = \lambda y.K'$ we must assume $y \sharp (x, J)$.

5. The datatype $\mathbb{D}$ of de Bruijn expressions

In this section we introduce the datatype $\mathbb{D}$ of *de Bruijn expressions*. Since we wish to relate $\mathbb{D}$ with $\mathbb{L}$, we will construct a larger domain $\mathbb{SD}$ of symbolic expressions which contains both $\mathbb{L}$ and $\mathbb{D}$ naturally, and study the structure of $\mathbb{SD}$.

We have formally checked all the lemmata, propositions and theorems in this section in Minlog.

5.1. *The datatype* $\mathbb{SD}$ *and its subset* LD

The datatype $\mathbb{SD}$ is defined inductively as shown in Fig. 8. We call an element in $\mathbb{SD}$ an $\mathbb{SD}$-expression.

Notational convention 5. We use X, Y, Z as metavariables ranging over $\mathbb{SD}$-expressions. We write $(X\ Y)$ and also XY for $\mathsf{sdapp}(X, Y)$. We will write $m\backslash X$ for $\mathsf{sdmask}(m, X)$, and write $[X]$ for $\mathsf{sdbind}(X)$.

$$\frac{}{x \in \mathbb{SD}}\ par \qquad \frac{}{\Box \in \mathbb{SD}}\ box \qquad \frac{}{i \in \mathbb{SD}}\ idx$$

$$\frac{X \in \mathbb{SD} \quad Y \in \mathbb{SD}}{XY \in \mathbb{SD}}\ \mathsf{sdapp} \qquad \frac{m \in \mathbb{M} \quad X \in \mathbb{SD}}{m\backslash X \in \mathbb{SD}}\ \mathsf{sdmask} \qquad \frac{X \in \mathbb{SD}}{[X] \in \mathbb{SD}}\ \mathsf{sdbind}$$

Fig. 8. Definition of the datatype $\mathbb{SD}$.

We define the *mask degree* $md(X)$, the *bind degree* $bd(X)$, and the *abstraction degree* $ad(X)$ of an $\mathbb{SD}$-expression:

Definition 10 (*Mask Degree, Bind Degree, Abstraction Degree*)**.**

$$\begin{array}{rcl@{\qquad}rcl} md & : & \mathbb{SD} \to \mathbb{I} & bd & : & \mathbb{SD} \to \mathbb{I} \\ md(X) & := & 0 \text{ if } X \in \mathbb{X} \cup \{\Box\} \cup \mathbb{I} & bd(X) & := & 0 \text{ if } X \in \mathbb{X} \cup \{\Box\} \cup \mathbb{I} \\ md(XY) & := & md(X) + md(Y) & bd(XY) & := & bd(X) + bd(Y) \\ md(m\backslash X) & := & md(X) + 1 & bd(m\backslash X) & := & bd(X) \\ md([X]) & := & md(X) & bd([X]) & := & bd(X) + 1 \end{array}$$

$$ad(X) := md(X) + bd(X).$$

We also define the set $\mathrm{FI}(X)$ of *free indices* in X as follows.

$$\begin{array}{rcl} \mathrm{FI}(X) & := & \{\} \quad \text{if } X \in \mathbb{X} \text{ or } X = \Box \\ \mathrm{FI}(i) & := & \{i\} \\ \mathrm{FI}(XY) & := & \mathrm{FI}(X) \cup \mathrm{FI}(Y) \\ \mathrm{FI}(m\backslash X) & := & \mathrm{FI}(X) \\ \mathrm{FI}([X]) & := & \{i - 1 \mid i \in \mathrm{FI}(X) \text{ and } i > 0\}. \end{array}$$

An $\mathbb{SD}$-expression is *index closed* if $\mathrm{FI}(X) = \{\}$.

Example 1. The abstracts constructor sdbind abstracts an index i occurring in its argument X by counting the number k of *sdbound* abstracts having the occurrence of i in its scope. (We do not count the *sdmasked* abstracts.) Then the occurrence of i becomes bound if $k = i + 1$. For example, the S combinator $\mathsf{lam}(x, \mathsf{lam}(y, \mathsf{lam}(z, (xz\ yz))))$, is represented by [[[(20 10)]]]. Using the sdmask constructor, S is represented by $(10\ 00)\backslash(00\ 10)\backslash(01\ 01)\backslash(\Box\Box\ \Box\Box)$. Both of these representations are LD-expressions since they are both index closed. We will see that we can *toggle* between these representations by the *toggle* function we introduce in Definition 13.

We next define the subsets LD_i ($i \in \mathbb{I}$) of $\mathbb{SD}$ inductively as in Fig. 9. It is easy to see that if $X \in \mathsf{LD}_i$ and $j \in \mathrm{FI}(X)$, then $j < i$, and that if $X \in \mathsf{LD}_i$ and $i < j$, then $X \in \mathsf{LD}_j$. We also note that $\mathsf{L}_i \subsetneq \mathsf{LD}_i$ and $\mathsf{D}_i \subsetneq \mathsf{LD}_i$.

We call an element in LD_i an LD_i-expression. Note that we first define the divisibility relation between maps and $\mathbb{SD}$-expressions. We then define the subset LD of $\mathbb{SD}$ by putting $\mathsf{LD} := \mathsf{LD}_0$. Note that if $X \in \mathsf{LD}$, then X is index-closed. We call an element of LD an LD-*expression*.

$$\frac{X \in \mathbb{X} \cup \{\Box\} \cup \mathbb{I}}{0 \mid X} \qquad \frac{}{1 \mid \Box} \qquad \frac{m \mid X \quad n \mid Y}{mn \mid XY}$$

$$\frac{m \mid X \quad n \mid X \quad m \perp n}{m \mid n\backslash X} \qquad \frac{m \mid X}{m \mid [X]}$$

$$\frac{}{x \in \mathsf{LD}_i}\ par \qquad \frac{}{\Box \in \mathsf{LD}_i}\ box \qquad \frac{j < i}{j \in \mathsf{LD}_i}\ idx \qquad \frac{X \in \mathsf{LD}_i \quad Y \in \mathsf{LD}_i}{XY \in \mathsf{LD}_i}\ \mathsf{app}$$

$$\frac{m \in \mathbb{M} \quad X \in \mathsf{LD}_i \quad m \mid X}{m\backslash X \in \mathsf{LD}_i}\ \mathsf{mask} \qquad \frac{X \in \mathsf{LD}_{i+1}}{[X] \in \mathsf{LD}_i}\ \mathsf{bind}$$

Fig. 9. Definitions of the divisibility relation and LD.

We single out some meaningful subsets of $\mathbb{SD}$ as follows.

$$\begin{aligned}
\mathsf{L}_i &:= \{X \in \mathsf{LD}_i \mid bd(X) = 0\} \quad (i \in \mathbb{I}) \\
\mathsf{L} &:= \mathsf{L}_0 \\
\mathbb{D} &:= \{X \in \mathbb{SD} \mid md(X) = 0\} \\
\mathsf{D}_i &:= \{X \in \mathsf{LD}_i \mid md(X) = 0\} \quad (i \in \mathbb{I}) \\
\mathsf{D} &:= \mathsf{D}_0.
\end{aligned}$$

Here, we have the following inclusion relations.

$$\begin{aligned}
\mathsf{L} &= \mathsf{L}_0 \subsetneq \mathsf{L}_1 \subsetneq \mathsf{L}_2 \cdots \\
\mathsf{D} &= \mathsf{D}_0 \subsetneq \mathsf{D}_1 \subsetneq \mathsf{D}_2 \cdots \subsetneq \mathbb{D}.
\end{aligned}$$

We can easily see that the set L is isomorphic to the structure of the symbolic lambda expressions we introduced in Section 3.1, since an element in L is index closed and it is constructed without using the bind function. (Note that by identifying these isomorphic sets, by abuse of the language, we are using the same notation L for these sets.) For this reason, we will call an element of L a *lambda expression*. Similarly, the set D becomes isomorphic to the structure of the *locally nameless* λ terms studied in, e.g., Aydemir et al. [1], since an element in D is index closed and it is constructed without using the *mask* function. (Strictly speaking, terms in [1] do not have $\Box$, but this is not essential.) For this reason, we will call an element of D a *locally nameless de Bruijn-expression*. On the other hand, an element in $\mathbb{D}$ is not index closed in general, so we will call it a *de Bruijn-expression*.

Remark 3. We will be only interested in the set LD and will work in it from now on. We remark that we had to define an infinite family of sets LD_i ($i \in \mathbb{I}$) to define $\mathsf{LD} = \mathsf{LD}_0$ because of the rule bind. By the same token, even though we are only interested in LD, when we prove properties of $X \in \mathsf{LD}$, we usually have to generalize the properties and prove them for arbitrary $X \in \mathsf{LD}_i$ ($i \in \mathbb{I}$).

In general, the mask degree and the bind degree of an LD-expression X can be both positive, and this means that X has characteristics of both lambda expression and de Bruijn-expression.

Suggested by this observation, we introduce a basic *toggle function* (Definition 13) which toggles the states of all the abstraction nodes $X \in \mathbb{SD}$. Namely if a node in X is a masking node then it will be changed to a binding node and vice versa.

We also define the following three auxiliary functions

$$\begin{array}{ll} \mathit{Out} & : \mathbb{I} \times \mathbb{M} \times \mathbb{SD} \to \mathbb{SD}, \\ \mathit{Map} & : \mathbb{I} \times \mathbb{SD} \to \mathbb{M}, \quad \text{and} \\ \mathit{Skel} & : \mathbb{I} \times \mathbb{SD} \to \mathbb{SD} \end{array}$$

on the way. We will see that *Out* and *Map*/*Skel* are inverses to each other (Proposition 18.8–14).

Definition 11 (*Out Function*)**.** We define *Out* inductively as follows. We write X^i_m for $\mathit{Out}_i(m, X)$. *Out* is a partial function, since X^i_m is defined only if $m \mid X$. X^i_m changes every $\Box$ in X bound by m into $i +$ (its binding height in X).

$$\begin{aligned} X^i_0 &:= X \quad \text{if } X \in \mathbb{X} \cup \{\Box\} \\ j^i_0 &:= \begin{cases} j & \text{if } j < i, \\ j+1 & \text{otherwise.} \end{cases} \\ \Box^i_1 &:= i \\ (X\ Y)^i_{mn} &:= (X^i_m\ Y^i_n) \\ (n\backslash X)^i_m &:= n\backslash(X^i_m) \\ [X]^i_m &:= [X^{i+1}_m]. \end{aligned}$$

Definition 12 (*Map and Skeleton*)**.** We define *Map* and *Skel* inductively as follows. We write X_i for $\mathit{Map}_i(X)$, and X^i for $\mathit{Skel}_i(X)$. Every index $i +$ (its binding height in X) is changed into 1 by X_i and into $\Box$ by X^i.

$$\begin{aligned} X_i &:= 0 \quad \text{if } X \in \mathbb{X} \cup \{\Box\} & X^i &:= X \quad \text{if } X \in \mathbb{X} \cup \{\Box\} \\ j_i &:= \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} & j^i &:= \begin{cases} j & \text{if } j < i, \\ \Box & \text{if } j = i, \\ j-1 & \text{if } j > i. \end{cases} \\ (X\ Y)_i &:= (X_i\ Y_i) & (X\ Y)^i &:= (X^i\ Y^i) \\ (m\backslash X)_i &:= X_i & (m\backslash X)^i &:= m\backslash X^i \\ [X]_i &:= X_{i+1} & [X]^i &:= [X^{i+1}]. \end{aligned}$$

The following lemma on *Map* can be easily shown.

Lemma 17 (*Simple Properties of Map*)**.**

1. $X \in \mathsf{LD}_i, 1 \mid X \Longrightarrow X_j = 0$.
2. $X \in \mathsf{LD}_i, m \mid X \Longrightarrow X_j \perp m$.

We will use the following properties of *Out*, *Map* and *Skel* to establish the key properties of the toggle function. Proposition 18.13 and 14 show that *Out* and *Skel* are inverses to each other.

Proposition 18 (*Properties of Out*, *Map and Skel*)**.**

1. $X \in \mathsf{LD}_i, j \le i, m \mid X, n \mid X, m \perp n \Longrightarrow m \mid X^j_n$.
2. $X \in \mathsf{LD}_i, j \le i, m \mid X \Longrightarrow X^j_m \in \mathsf{LD}_{i+1}$.

3. $m \mid X \Longrightarrow X_j \perp m$.
4. $m \mid X \Longrightarrow m \mid X^j$.
5. $X \in \mathsf{LD}_{i+1}, j \leq i \Longrightarrow X^j \in \mathsf{LD}_i$.
6. $X \in \mathsf{LD}_i \Longrightarrow X_j \mid X^j$.
7. $X \in \mathsf{LD}_i, m \mid X, n \mid X, m \perp n, k \leq j \Longrightarrow (X_n^j)_m^k = (X_m^k)_n^{j+1}$.
8. $X \in \mathsf{LD}_i, m \mid X, k \leq j \Longrightarrow (X_m^k)_{j+1} = X_j$.
9. $X \in \mathsf{LD}_i, m \mid X, k \leq j \Longrightarrow (X_m^{j+1})_k = X_k$.
10. $X \in \mathsf{LD}_i, m \mid X, k \leq j \Longrightarrow (X_m^{j+1})^k = (X^k)_m^j$.
11. $X \in \mathsf{LD}_i, m \mid X, k \leq j \Longrightarrow (X_m^k)^{j+1} = (X^j)_m^k$.
12. $X \in \mathsf{LD}_i, m \mid X, j \leq i \Longrightarrow (X_m^j)_j = m$.
13. $X \in \mathsf{LD}_i, m \mid X, j \leq i \Longrightarrow (X_m^j)^j = X$.
14. $X \in \mathsf{LD}_{i+1}, j \leq i \Longrightarrow (X^j)^j_{(X_j)} = X$.
15. $X \in \mathsf{LD}_i, i \leq j \Longrightarrow X^j = X$.
16. $X \in \mathsf{LD}_i, k \leq j \Longrightarrow (X^k)_j = X_{j+1}$.
17. $X \in \mathsf{LD}_i, k \leq j \Longrightarrow (X^{j+1})_k = X_k$.
18. $X \in \mathsf{LD}_i, k \leq j \Longrightarrow (X^{j+1})^k = (X^k)^j$.
19. $m \mid X \Longrightarrow bd(X_m^j) = bd(X)$ *and* $md(X_m^j) = md(X)$.
20. $bd(X^j) = bd(X)$ *and* $md(X^j) = md(X)$.

Proof. See the Appendix. □

Definition 13 (*toggle Function*)**.** We define *toggle* : $\mathbb{SD} \to \mathbb{SD}$ inductively as follows. It is a partial function since it uses the *Out* function. We write $\langle X \rangle$ for $toggle(X)$.

$$\begin{aligned} \langle X \rangle &:= X \quad \text{if } X \in \mathbb{X} \cup \{\Box\} \cup \mathbb{I} \\ \langle XY \rangle &:= \langle X \rangle \langle Y \rangle \\ \langle m \backslash X \rangle &:= [\langle X \rangle_m^0] \\ \langle [X] \rangle &:= \langle X \rangle_0 \backslash \langle X \rangle^0. \end{aligned}$$

Although *toggle* is a partial function on $\mathbb{SD}$, its restriction to LD_i ($i \in \mathbb{I}$) is total as we see now. As *toggle* changes a masking node to a binding and vice versa (using *Out* and *Skel* which are inverses to each other) and leaves an application node and an atomic node unchanged, it is intuitively clear that we have $\langle \langle X \rangle \rangle = X$. However its proof is subtle because we have to manipulate de Bruijn indices carefully while computing $\langle X \rangle$.

Proposition 19 (*Properties of the toggle Function*)**.**

1. $X \in \mathsf{LD}_i \Longrightarrow \langle X \rangle \in \mathsf{LD}_i$.
2. $X \in \mathsf{LD}_i, m \mid X \Longrightarrow m \mid \langle X \rangle$.
3. $X \in \mathsf{LD}_i \Longrightarrow bd(\langle X \rangle) = md(X)$.
4. $X \in \mathsf{LD}_i \Longrightarrow md(\langle X \rangle) = bd(X)$.
5. $X \in \mathsf{LD}_i \Longrightarrow ad(\langle X \rangle) = ad(X)$.
6. $X \in \mathsf{LD}_i, m \mid X \Longrightarrow \langle X_m^j \rangle = \langle X \rangle_m^j$.
7. $X \in \mathsf{LD}_i \Longrightarrow \langle X \rangle_j = X_j$.
8. $X \in \mathsf{LD}_i \Longrightarrow \langle X^j \rangle = \langle X \rangle^j$.
9. $X \in \mathsf{LD}_i \Longrightarrow \langle \langle X \rangle \rangle = X$.

Proof. See the Appendix. □

Example 2. We illustrate some instances of Proposition 19. Consider the raw lambda term $K = \lambda x.(\lambda y.\, yx)x$. K can be represented in LD by the four expressions as shown in the table below.

X	*md*	*bd*	L?	D?	$\langle X\rangle_x$	$\langle X\rangle_y$	$\langle X\rangle$
$X_a = (01\ 1)\backslash(10\backslash(\square\ \square)\ \square)$	2	0	Yes	No	X_b	X_c	X_d
$X_b = [(10\backslash(\square\ 0)\ 0)]$	1	1	No	No	X_a	X_d	X_c
$X_c = (01\ 1)\backslash([0\square]\ \square)$	1	1	No	No	X_d	X_a	X_b
$X_d = [([01]\ 0)]$	0	2	No	Yes	X_c	X_b	X_a

In K, the outer abstraction is done by abstracting the parameter x and the inner by abstracting y. In LD two abstraction functions mask and bind are available, and we can freely choose any of these when we make abstracts. So we have four different ways of representing K in LD. The *md* and *bd* columns show the number of times these functions are used to construct the four representations.

In general, if a raw lambda term L has k abstracts in it, it can be represented in LD in 2^k ways. Of these only one belongs to L and another one belongs to D as can be seen in the columns 'L?' and 'D?'.

The toggle function $\langle X\rangle$ is *global* in the sense that it toggles all the k abstraction nodes in L in one shot. It is also possible to define *local* toggle functions which toggle only a specified abstraction node in L. In the table above, we showed the effects of these local toggle functions $\langle X\rangle_x$ (toggles the node which corresponds to the binding node x in K) and $\langle X\rangle_y$ (toggles the y node) as well as the global toggle function $\langle X\rangle$. Since each local toggle affects only one binding node, in case of K, the global toggle can be realized by composing the two local toggle functions in any order. Namely, we have

$$\langle X\rangle = \langle\langle X\rangle_x\rangle_y = \langle\langle X\rangle_y\rangle_x$$

as can be seen in the rightmost three columns.

We show graphical representations of these expressions in Fig. 10. In the center of the figure we put the picture of the representation of K which is obtained by the method first introduced by Quine [15] and later by Bourbaki [3]. The central picture is obtained from K by replacing the binding occurrences of parameters with white circles and the bound occurrences with black circles and at the same time connecting the corresponding bind/bound circles by the three lines shown in the figure. Quine calls these lines *bonds*. It is intuitively clear that this method always gives a correct canonical representation of any raw lambda term. But, the problem with this approach is the difficulty of giving a formal inductive definition of the representations.

The four pictures surrounding the central picture are graphical representations of the four expressions shown in the table. In these pictures, unlike bonds in the Quine–Bourbaki notation, each bond has a direction either from a binding node to a bound node (left to right) or from a bound node to a binding node (right to left). The direction of a bond shows which of mask or bind is used to construct the abstract which correspond to the binding node of the bond. Note that each black circle corresponds to a box in case of mask and to an index in case of bind. The figure also shows the commutation of the two local toggle functions $\langle X\rangle_x$ and $\langle X\rangle_y$.

5.2. Bijective correspondence between L *and* D

In this subsection, we will show that there is a natural bijective correspondence between L and D which respects substitution.

We can obtain the following theorem as a corollary to Proposition 18.{3, 4, 9}.

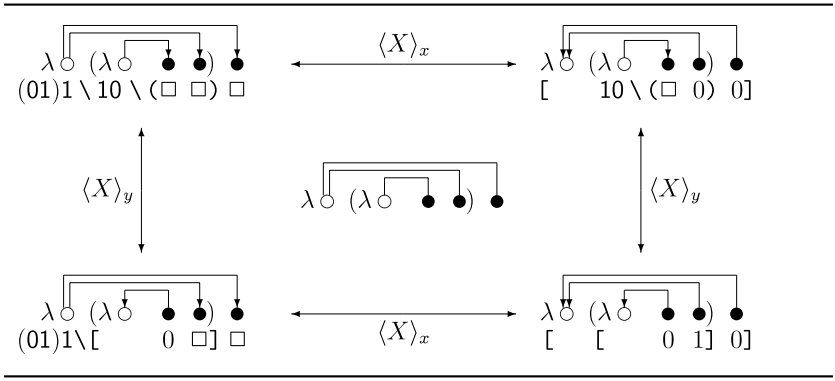

Fig. 10. Representations of $\lambda x.\,(\lambda y.\,yx)x$ in LD and in Quine–Bourbaki notation.

Theorem 20 (*Bijections Between* L_i *and* D_i)**.**

1. *toggle restricted to* L_i *is a bijection from* L_i *to* D_i.
2. *toggle restricted to* D_i *is a bijection from* D_i *to* L_i.

By this theorem we see that L and D are bijectively related by the *toggle* function. We will now see moreover that L and D respect the substitution operation with each other. In order to show this, we first define the *substitution* function on $\mathbb{SD}$ as follows.

Definition 14 (*Substitution on* $\mathbb{SD}$)**.**

$$
\begin{aligned}
y\{x\backslash X\} &:= \begin{cases} X & \text{if } x = y, \\ y & \text{if } x \neq y. \end{cases} \\
\Box\{x\backslash X\} &:= \Box \\
(Y\ Z)\{x\backslash X\} &:= (Y\{x\backslash X\}\ Z\{x\backslash X\}) \\
(m\backslash Y)\{x\backslash X\} &:= m\backslash(Y\{x\backslash X\}) \\
i\{x\backslash X\} &:= i \\
[Y]\{x\backslash X\} &:= [Y\{x\backslash X\}].
\end{aligned}
$$

Remark 4. In Section 3 we defined substitution (Definition 4) in terms of the instantiation operation which, in turn, is defined in terms of the hole filling operation. Here, we define substitution directly by structural recursion. We take this approach since (1) substitution defined here restricted to L is the same as substitution defined in Definition 4 thanks to the first four equations of Proposition 1 and (2) substitution defined here restricted to D is the same as substitution defined in Fig. 2 of Aydemir et al. [1].

Lemma 21 (*Properties of Substitution*)**.**

1. $X \in \mathsf{LD},\ Y \in \mathsf{LD}_i, m \mid Y \Longrightarrow Y\{x\backslash X\}^k_m = Y^k_m\{x\backslash X\}$.
2. $X \in \mathsf{LD},\ Y \in \mathsf{LD}_{i+1}, k \leq i \Longrightarrow Y\{x\backslash X\}_k = Y_k$.
3. $X \in \mathsf{LD},\ Y \in \mathsf{LD}_{i+1}, k \leq i \Longrightarrow Y\{x\backslash X\}^k = Y^k\{x\backslash X\}$.

Proof. By induction on Y. We will deal only with interesting cases.

Proof of 1. The case where $Y = n\backslash Z$. We show the conclusion by assuming that $Y \in \mathsf{LD}_i$ and $m \mid Y$. In this case we have $n\backslash Z \in \mathsf{LD}_i$, namely, $Z \in \mathsf{LD}_i$ and $n \mid Z$. Also, since $m \mid Y$,

we have $m \mid Z$ and $n \perp m$. We have to show $(n\backslash Z)\{x\backslash X\}^k_m = (n\backslash Z)^k_m\{x\backslash X\}$, or equivalently, $Z\{x\backslash X\}^k_m = Z^k_m\{x\backslash X\}$. We can show this by IH.

Proof of 2 and *proof of* 3 are easy. □

Using the above lemma, we can prove the following proposition.

Proposition 22 (*toggle Preserves Substitution*)**.**

$$X \in \mathsf{LD},\ Y \in \mathsf{LD}_i \Longrightarrow \langle Y\{x\backslash X\}\rangle = \langle Y\rangle\{x\backslash\langle X\rangle\}.$$

Proof. Induction on Y.

- The case where Y is $n\backslash Z$. In this case we have $n\backslash Z \in \mathsf{LD}_i$, namely, $Z \in \mathsf{LD}_i$ and $n \mid Z$. We have to show $\langle(n\backslash Z)\{x\backslash X\}\rangle = \langle n\backslash Z\rangle\{x\backslash\langle X\rangle\}$, or equivalently, $\langle Z\{x\backslash X\}\rangle^0_n = \langle Z\rangle^0_n\{x\backslash\langle X\rangle\}$. We can show this as follows.

$$\begin{aligned}\langle Z\{x\backslash X\}\rangle^0_n &= \langle Z\rangle\{x\backslash\langle X\rangle\}^0_n && \text{by IH}\\ &= \langle Z\rangle^0_n\{x\backslash\langle X\rangle\} && \text{by Lemma 21.1, Proposition 19.}\{1,2\}.\end{aligned}$$

- The case where Y is $[Z]$. We have to show $\langle[Z]\{x\backslash X\}\rangle = \langle[Z]\rangle\{x\backslash\langle X\rangle\}$, or equivalently,

$$(1)\ \langle Z\{x\backslash X\}\rangle_0 = \langle Z\rangle_0 \quad \text{and} \quad (2)\ \langle Z\{x\backslash X\}\rangle^0 = \langle Z\rangle^0\{x\backslash\langle X\rangle\}.$$

We have (1) as follows.

$$\begin{aligned}\langle Z\{x\backslash X\}\rangle_0 &= \langle Z\rangle\{x\backslash\langle X\rangle\}_0 && \text{by IH}\\ &= \langle Z\rangle_0 && \text{by Lemma 21.2.}\end{aligned}$$

We have (2) as follows.

$$\begin{aligned}\langle Z\{x\backslash X\}\rangle^0 &= \langle Z\rangle\{x\backslash\langle X\rangle\}^0 && \text{by IH}\\ &= \langle Z\rangle^0\{x\backslash\langle X\rangle\} && \text{by Lemma 21.3.}\ \square\end{aligned}$$

As a special case of this proposition, we have the following theorem.

Theorem 23 (*toggle Preserves Substitution on* D *and* L)**.**

1. $X \in \mathsf{D},\ Y \in \mathsf{D} \Longrightarrow \langle Y\{x\backslash X\}\rangle = \langle Y\rangle\{x\backslash\langle X\rangle\} \in \mathsf{L}$.
2. $X \in \mathsf{L},\ Y \in \mathsf{L} \Longrightarrow \langle Y\{x\backslash X\}\rangle = \langle Y\rangle\{x\backslash\langle X\rangle\} \in \mathsf{D}$.

Combining Theorem 23.1 with Theorem 20.2 we see that L correctly represents D respecting substitution.

5.3. Commutation of toggle and β-conversion

Apart from substitution for parameters in L and in D we can also consider the respective β-conversion rules, β_{L} and β_{D}. It turns out that *toggle* commutes with β-conversion on either side. First we need to define instantiation on $\mathbb{SD}$, to be able to even formulate β_{D}.

Definition 15 (*Instantiation on* $\mathbb{SD}$)**.**

$$\begin{aligned}X\triangledown_j Y &:= X \quad \text{for } X = x \text{ or } X = \square\\ (X_1\ X_2)\triangledown_j Y &:= (X_1\triangledown_j Y\ X_2\triangledown_j Y)\\ (m\backslash X)\triangledown_j Y &:= X\triangledown_j Y\\ i\triangledown_j Y &:= \begin{cases} i & \text{if } i < j\\ Y & \text{if } i = j\\ i-1 & \text{if } i > j.\end{cases}\\ [X]\triangledown_j Y &:= [X\triangledown_{j+1} Y].\end{aligned}$$

Clearly we need an index shift here when we move under a bind. We defined this operation on the entire $\mathbb{SD}$ structure, but will only use it on the D-part.

We first show that commutation holds when we start with a β_{D}-redex.

$$\begin{array}{ccc} (\langle X\rangle_0 \backslash \langle X\rangle^0)\langle Y\rangle & \xleftarrow{\quad toggle \quad} & [X]Y \\ \downarrow{\scriptstyle \beta_{\mathsf{L}}} & & \downarrow{\scriptstyle \beta_{\mathsf{D}}} \\ \langle X\rangle^0{}_{\langle X\rangle_0}[\langle Y\rangle] = \langle X \triangledown_0 Y\rangle & \xleftarrow{\quad toggle \quad} & X \triangledown_0 Y \end{array}$$

For the proof we need an iteration *ItSkel* of the skeleton function

$$\begin{aligned} \mathit{ItSkel}_0(X) &:= X, \\ \mathit{ItSkel}_{i+1}(X) &:= \mathit{ItSkel}_i(X^i) \end{aligned}$$

and a lemma

$$\langle \mathit{ItSkel}_j(X \triangledown_j Y)\rangle = \langle \mathit{ItSkel}_{j+1}(X)\rangle_{X_j}[\langle Y\rangle],$$

assuming $md(X) = md(Y) = 0$ and that $Y \in \mathsf{LD}$.

Next we show that *toggle* and β commute when we start with a β_{L}-redex, i.e., from $(m\backslash X)Y$. We need to assume $X, Y \in \mathsf{LD}$ with $bd(X) = bd(Y) = 0$, and $m \mid X$.

$$\begin{array}{ccc} (m\backslash X)Y & \xrightarrow{\quad toggle \quad} & [\langle X\rangle^0_m]\langle Y\rangle \\ \downarrow{\scriptstyle \beta_{\mathsf{L}}} & & \downarrow{\scriptstyle \beta_{\mathsf{D}}} \\ X_m[Y] & \xrightarrow{\quad toggle \quad} & \langle X_m[Y]\rangle = \langle X\rangle^0_m \triangledown_0 \langle Y\rangle \end{array}$$

This will follow from the previous commutative diagram when we instantiate it with $X \mapsto \langle X\rangle^0_m$ and $Y \mapsto \langle Y\rangle$. Hence we have

$$\langle \langle X\rangle^0_m\rangle^0_{\langle\langle X\rangle^0_m\rangle_0}[\langle\langle Y\rangle\rangle] = \langle \langle X\rangle^0_m \triangledown_0 \langle Y\rangle\rangle.$$

Now $\langle\langle X\rangle^0_m\rangle^0$ is the same as $\langle X\rangle$ by Proposition 18.13, since $X \in \mathsf{LD}$ and $m \mid X$ by assumption. Also $\langle\langle X\rangle^0_m\rangle_0$ is the same as m by Proposition 18.12, for the same reasons. Finally $\langle\langle Y\rangle\rangle$ is the same as Y by Proposition 19.9. Therefore we have

$$\langle X\rangle_m[Y] = \langle \langle X\rangle^0_m \triangledown_0 \langle Y\rangle\rangle.$$

One application of *toggle* gives the claim.

6. The $\mathsf{L}_{\beta\eta}$ calculus

In this section we develop the $\lambda\beta\eta$ calculus, $\mathsf{L}_{\beta\eta}$, within L. The $\mathsf{L}_{\beta\eta}$-reduction rules are shown in Fig. 11. Using the isomorphism between L and nominal lambda calculus in Isabelle/HOL

$$\frac{}{AM \to_{\beta\eta} A \triangledown M}\ \beta \qquad \frac{}{(01\backslash M\square) \to_{\beta\eta} M}\ \eta$$

$$\frac{M \to_{\beta\eta} M'}{MN \to_{\beta\eta} M'N}\ \mathsf{appl} \qquad \frac{N \to_{\beta\eta} N'}{MN \to_{\beta\eta} MN'}\ \mathsf{appr}$$

$$\frac{M \to_{\beta\eta} N}{lam(x, M) \to_{\beta\eta} lam(x, N)}\ \xi$$

Fig. 11. Definition of the $\beta\eta$-reduction rules.

outlined in Section 4.5, it is easy to show that our $\mathsf{L}_{\beta\eta}$-reduction agrees with reduction defined on nominal terms.

An interesting thing about these rules is the name-free presentation of rules β and η. The informal η rule

$$\frac{x \notin \mathrm{FP}(M)}{\lambda x.\,(M\ x) \to_{\beta\eta} M}\ \eta$$

requires the side condition $x \notin \mathrm{FP}(M)$. This is not a question of α equivalence, and even canonical representations, such as de Bruijn nameless terms and Sato canonical terms [14] require this side condition in the η rule. We avoid the side condition in the map representation since we have

$$\mathsf{lam}(x, Mx) = 01\backslash M\square \qquad \text{if } x \notin \mathrm{FP}(M).$$

Note that in the informal η rule, parameter x only occurs bound: the η rule is about abstracts, and has nothing to do with parameters. Our rule, parametric in M but not mentioning a name, captures this observation.

In the informal β rule [2]

$$\frac{}{(\lambda x.\,M)\ K \to_{\beta\eta} M\{x\backslash K\}}\ \beta$$

schematic parameter x is bound on the left hand side, and free on the right hand side of the rule. Thus, this rule must be read up to α equivalence, and we must choose a concrete α-representative of $\lambda x.\,M$ to apply it. With the map representation we are able to write rule β in name-free form as shown in Fig. 11.

6.1. A fly in the ointment

It is only luck that we can write rule β in name-free form (because the left hand and right hand sides of the rule work on the same abstraction, A in Fig. 11). Many rules that we want to write are not obviously expressible in name-free form, e.g. rule ξ in Fig. 11. (We do not exclude that new ideas may solve this problem.) This raises two questions: why are some rules not expressible in name-free form? And why does it matter?

6.1.1. Why don't we write rule ξ in name-free form?

Perhaps you conjecture that the following could be used for rule ξ:

$$\frac{M \to_{\beta\eta} N}{m\backslash M \to_{\beta\eta} m\backslash N}\ \ddots \tag{2}$$

Unfortunately not, as the following instance of this putative rule ξ shows:

$$\frac{((1\backslash\Box)\ \Box) \to_{\beta\eta} \Box}{(0\ 1)\backslash((1\backslash\Box)\ \Box) \to_{\beta\eta} (0\ 1)\backslash\Box}\ .$$

In the conclusion of this "rule", $(0\ 1)\backslash\Box$ is not even a well-formed L term. (The correct RHS is $1\backslash\Box$.)

For named λ terms the operation of substitution leaves the free names (the indicators of positions which may still be bound) unchanged. Similarly with de Bruijn nameless terms, substitution leaves the free indexes (as viewed from outside the term) unchanged. However with the L representation, the indicators of positions which may still be bound do not occur in the terms themselves; they are the maps that divide the term (maps dividing the base term, combined in a complicated but functional way with the maps dividing the term being implanted). Further since the shapes of terms change under substitution, it is clear that the set of indicators of positions which may still be bound is not preserved by substitution. Thus it is impossible to have the same map abstracted on both sides of the conclusion of a correct ξ rule, as in Eq. (2). The excursion through names in rule ξ of Fig. 11 serves to compute the appropriate maps indicating the positions to be bound in the conclusion of the rule. (Recall from Definition 4 that *lam* is a defined function that computes an L term.) If one wants a ξ rule with a name-free conclusion it is the following:

$$\frac{x \sharp (M, N) \quad M_m[x] \to_{\beta\eta} N_n[x]}{m\backslash M \to_{\beta\eta} n\backslash N}\ . \tag{3}$$

Similar to rule ξ of Fig. 11, this rule mentions a free parameter in the premise. This rule is also reminiscent of the representations discussed in [11,1,19].

It is not only rule ξ that poses this problem for the map representation. For another naturally occurring example that appears to require use of parameters, consider rule β of Tait/Martin–Löf parallel reduction.

6.1.2. Why does it matter that rule ξ uses names?

One of the goals of our map representation is to avoid the need to reason by equivariance and permutation of names that seems necessary in representations using names [11,1,19,23]. In Lemma 2 we showed that the usual substitution lemma of λ calculus can be proved in our notation by term induction, without the usual α-converting to fresh names. Analogously for $\to_{\beta\eta}$, try to prove

$$M_1 \to_{\beta\eta} M_2 \implies M_1\{x\backslash N\} \to_{\beta\eta} M_2\{x\backslash N\}$$

by rule induction on $M_1 \to_{\beta\eta} M_2$. In the case for rule ξ, where $M_1 = \lambda y.\, P$ you will need to α-convert M_1 so that $y \sharp (x, N)$, allowing the substitution to go under the binder so the induction hypothesis can be used.

The problem with rule ξ is that there are (infinitely) many instances of the rule with the same conclusion. e.g. the two instances

$$\frac{(1\backslash\Box)x \to_{\beta\eta} x}{lam(x, (1\backslash\Box)x) \to_{\beta\eta} lam(x, x)} \qquad \frac{(1\backslash\Box)y \to_{\beta\eta} y}{lam(y, (1\backslash\Box)y) \to_{\beta\eta} lam(y, y)}$$

have equal conclusions but distinct premises. When we say "by rule induction on $M_1 \to_{\beta\eta} M_2$" we are destructing a hypothetical derivation, and this derivation contains some α-variant of $\lambda y.\, P$

in the conclusion of (hypothetical uses of) rule ξ and an instance of P with free y in the premise. But we do not know (and cannot specify) which α-variant of $\lambda y.\ P$ occurs; it is not visible in the judgement $M_1 \to_{\beta\eta} M_2$ whose derivation we are destructing. The usual solution is to reason locally by equivariance, or to package such reasoning in a derivable induction rule [11,1,19,23]. If we could write rule ξ in name-free form, we could avoid this digression in reasoning.

So is there a name-free rule ξ for our L representation (and similarly, name-free definitions for other relations on L, such as Tait/Martin–Löf parallel reduction)? We conjecture this is possible, and leave it for future work.[3] Just as the name-free notion of hole filling led to a proof of the substitution lemma without choosing fresh names (Lemma 2), we hope that a name-free definition of β-reduction would lead to proofs by rule induction and equational reasoning only.

7. Conclusion

We have presented a canonical, name-free representation of lambda terms and proved it to be an adequate representation with respect to both the nominal logic representation and pure de Bruijn representation. These proofs are formalized in Isabelle/HOL and Minlog respectively. We have used our representation as a lens to examine both raw lambda syntax and the well-known de Bruijn nameless representation of binding.

Among the technical results of our work is a proof of the substitution lemma of lambda calculus (Lemma 2) that proceeds by induction and pure equational reasoning, without any renaming. We have also given a definition of α equivalence for raw λ syntax that can be decided without any renaming.

We present a definition of $\beta\eta$-reduction for our representation (not quite name-free), and discuss how it might be made name-free by future work, and what that might buy.

Acknowledgments

Masahiko Sato was supported by JSPS KAKENHI Grant Number 22300008 and Takafumi Sakurai was supported by JSPS KAKENHI Grant Number 24650002. Randy Pollack was supported by the DARPA CRASH program through the US Air Force Research Laboratory under Contract No. FA8650-10-C-7090. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the US Government.

Appendix. Proofs

A.1. Proofs in Section 4

Proof of Lemma 8. By induction on the size of K and L, followed by case analysis on the derivation of $K^{x/\!\!/z} =_\alpha L^{y/\!\!/z}$.Consider the case that the last rule of the derivation is the lam-rule (other cases are easy); so let $\mathsf{lam}(z_1, K)^{x/\!\!/z} =_\alpha \mathsf{lam}(z_2, L)^{y/\!\!/z}$ be the conclusion of the last rule. We have 2×2 cases: ($x = z_1$ or $x \neq z_1$) and ($y = z_2$ or $y \neq z_2$).

In the case $x \neq z_1$ and $y \neq z_2$, the last rule has the following form.

$$\frac{(K^{x/\!\!/z})_{z_1} = (L^{y/\!\!/z})_{z_2} \quad (K^{x/\!\!/z})^{z_1} =_\alpha (L^{y/\!\!/z})^{z_2}}{\mathsf{lam}(z_1, K^{x/\!\!/z}) =_\alpha \mathsf{lam}(z_2, L^{y/\!\!/z})}$$

[3] James McKinna made a interesting suggestion towards this goal.

By the second premise and Lemma 7.1, we have $(K^{z_1})^{x/\!/z} =_\alpha (L^{z_2})^{y/\!/z}$. By IH, we have $(K^{z_1})^x =_\alpha (L^{z_2})^y$ and $(K^{z_1})_x = (L^{z_2})_y$. So, by Lemma 5.1, we have $(K^x)^{z_1} =_\alpha (L^y)^{z_2}$. On the other hand, we have $(K^x)_{z_1} = (L^y)_{z_2}$ by the first premise and Lemma 7.2. Therefore, we have $\mathsf{lam}(z_1, K^x) =_\alpha \mathsf{lam}(z_2, L^y)$, that is, $\mathsf{lam}(z_1, K)^x =_\alpha \mathsf{lam}(z_2, L)^y$. We also have $\mathsf{lam}(z_1, K)_x = \mathsf{lam}(z_2, L)_y$ by $(K^{z_1})_x = (L^{z_2})_y$ and Lemma 5.2.

In the case $x = z_1$ and $y \neq z_2$, the last rule has the following form.

$$\frac{(K^{x/\!/z})_z = (L^{y/\!/z})_{z_2} \quad (K^{x/\!/z})^z =_\alpha (L^{y/\!/z})^{z_2}}{\mathsf{lam}(z, K^{x/\!/z}) =_\alpha \mathsf{lam}(z_2, L^{y/\!/z})}$$

By the second premise and Lemma 7.{1, 3}, we have $(K^x)^{x/\!/z} =_\alpha (L^{z_2})^{y/\!/z}$. By IH, we have $(K^x)^x =_\alpha (L^{z_2})^y$ and $(K^x)_x = (L^{z_2})_y$. So, by Lemma 5.{1, 3}, we have $K^x =_\alpha (L^y)^{z_2}$. On the other hand, we have $K_x = (L^y)_{z_2}$ by the first premise and Lemma 7.{2, 3}. Therefore, we have $\mathsf{lam}(x, K) =_\alpha \mathsf{lam}(z_2, L^y)$, that is, $\mathsf{lam}(x, K)^x =_\alpha \mathsf{lam}(z_2, L)^y$. We also have $\mathsf{lam}(x, K)_x = 0 = \mathsf{lam}(z_2, L)_y$ by $(K^x)_x = (L^{z_2})_y$ and Lemma 5.{2, 3}.

Other cases are similar. □

Proof of Lemma 10. By induction on the size of K and L, followed by case analysis on the derivation of $K^x \sim L^y$.

(1) *par*-rule: Let $K^x \sim L^y$ be the conclusion of the rule. In this case, we have $K^x = L^y = z'$ for some parameter z'. Then, we have $K = z'$, $L = z'$, $x \neq z'$, and $y \neq z'$. We also have $z \neq x$ and $z \neq y$, so we have $K^{x/\!/z} \sim z' \sim L^{y/\!/z}$.

(2) *box*-rule: Let $K^x \sim L^y$ be the conclusion of the rule. In this case, we have $K^x = L^y = \Box$. Then, we have ($K = x$ or $K = \Box$) and ($L = y$ or $L = \Box$). Since $K_x = L_y$, we have ($K = x$ and $L = y$) or ($K = \Box$ and $L = \Box$). In both cases, we have $K^{x/\!/z} \sim L^{y/\!/z}$.

(3) app-rule: Easy.

(4) lam-rule: Let $\mathsf{lam}(z_1, K)^x \sim \mathsf{lam}(z_2, L)^y$ be the conclusion of the rule. We have 2×2 cases: ($x = z_1$ or $x \neq z_1$) and ($y = z_2$ or $y \neq z_2$).

In the case $x \neq z_1$ and $y \neq z_2$, the last rule has the following form.

$$\frac{(K^x)^{z_1/\!/z'} \sim (L^y)^{z_2/\!/z'} \quad z' \notin \{z_1, z_2\} \cup \mathrm{P}(K^x) \cup \mathrm{P}(L^y)}{\mathsf{lam}(z_1, K^x) \sim \mathsf{lam}(z_2, L^y)}$$

By equivariance of $\sim$, we can take z' so that z' satisfies $z' \notin \{z, x, y\}$ and the above condition since parameters are infinite. By the premise and Lemma 7.1, we have $(K^{z_1/\!/z'})^x \sim (L^{z_2/\!/z'})^y$. On the other hand, we have $K_x = L_y$ from $\mathsf{lam}(z_1, K)_x = \mathsf{lam}(z_2, L)_y$, so we have $(K^{z_1/\!/z'})_x = (L^{z_2/\!/z'})_y$ by Lemmas 5.2 and 7.2. Therefore, by IH, we have $(K^{z_1/\!/z'})^{x/\!/z} \sim (L^{z_2/\!/z'})^{y/\!/z}$. So, by the property of swap, we have $(K^{x/\!/z})^{z_1/\!/z'} \sim (L^{y/\!/z})^{z_2/\!/z'}$. Therefore, we have $\mathsf{lam}(z_1, K^{x/\!/z}) \sim \mathsf{lam}(z_2, L^{y/\!/z})$, that is, $\mathsf{lam}(z_1, K)^{x/\!/z} \sim \mathsf{lam}(z_2, L)^{y/\!/z}$.

In the case $x = z_1$ and $y \neq z_2$, the last rule has the following form.

$$\frac{K^{x/\!/z'} \sim (L^y)^{z_2/\!/z'} \quad z' \notin \{x, z_2\} \cup \mathrm{P}(K) \cup \mathrm{P}(L^y)}{\mathsf{lam}(x, K) \sim \mathsf{lam}(z_2, L^y)}$$

By equivariance of $\sim$, we can take z' so that z' satisfies $z' \notin \{z, y\}$ and the above condition since parameters are infinite. By the premise and Lemma 7.{1, 4}, we have $(K^{x/\!/z'})^x \sim (L^{z_2/\!/z'})^y$. On the other hand, we have $0 = L_y$ from $\mathsf{lam}(x, K)_x = \mathsf{lam}(z_2, L)_y$, so we have $(K^{x/\!/z'})_x = 0 = (L^{z_2/\!/z'})_y$ by Lemmas 5.2 and 7.{2, 4}. Therefore, by IH, we have $(K^{x/\!/z'})^{x/\!/z} \sim (L^{z_2/\!/z'})^{y/\!/z}$. So,

by the property of swap, we have $(K^{x/\!/z})^{z/\!/z'} \sim (L^{y/\!/z})^{z_2/\!/z'}$. Therefore, we have $\mathsf{lam}(z, K^{x/\!/z}) \sim \mathsf{lam}(z_2, L^{y/\!/z})$, that is, $\mathsf{lam}(x, K)^{x/\!/z} \sim \mathsf{lam}(z_2, L)^{y/\!/z}$.

Other cases are similar. □

Proof of Lemma 12. By induction on $|K|$. The crucial case is lam-case: given $\mathsf{lam}(z, K)$ and $y \notin \mathrm{FP}(\mathsf{lam}(z, K))$, the goal is to find L_0 such that $\mathsf{lam}(x, \mathsf{lam}(z, K)) =_\alpha \mathsf{lam}(y, L_0)$. We have two cases.

(1) $y \neq z$: In this case, we have $y \notin \mathrm{FP}(K)$. By IH, there exists L such that $\mathsf{lam}(x, K) =_\alpha \mathsf{lam}(y, L)$, that is, $K^x =_\alpha L^y$ and $K_x = L_y$. We have two subcases.

(1.1) $x \neq z$: By Lemma 6.2, we have $\mathsf{lam}(z, K^x) =_\alpha \mathsf{lam}(z, L^y)$. So, we have $\mathsf{lam}(z, K)^x = \mathsf{lam}(z, K^x) =_\alpha \mathsf{lam}(z, L^y) = \mathsf{lam}(z, L)^y$ and $\mathsf{lam}(z, K)_x = K_x = L_y = \mathsf{lam}(z, L)_y$. Therefore, $\mathsf{lam}(x, \mathsf{lam}(z, K)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(z, L))$.

(1.2) $x = z$: We have $\mathsf{lam}(x, K)^x = \mathsf{lam}(x, K) =_\alpha \mathsf{lam}(y, L) = \mathsf{lam}(y, L)^y$ and $\mathsf{lam}(x, K)_x = 0 = \mathsf{lam}(y, L)_y$. Therefore, $\mathsf{lam}(x, \mathsf{lam}(z, K)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(y, L))$.

(2) $y = z$: We can take z' such that $z' \notin \mathrm{FP}(K) \cup \{x, y\}$. By IH, there exists K' such that $\mathsf{lam}(z, K) =_\alpha \mathsf{lam}(z', K')$. Then, $|K| = |K'|$ and we have $y \notin \mathrm{FP}(K')$ from $\mathrm{FP}(K) - \{z\} = \mathrm{FP}(K') - \{z'\}$. Therefore, by IH, there exists L such that $\mathsf{lam}(x, K') =_\alpha \mathsf{lam}(y, L)$. Similarly to (1.1), we have $\mathsf{lam}(x, \mathsf{lam}(z', K')) =_\alpha \mathsf{lam}(y, \mathsf{lam}(z', L))$. Since $\mathsf{lam}(x, \mathsf{lam}(z, K)) =_\alpha \mathsf{lam}(x, \mathsf{lam}(z', K'))$ by Lemma 6.2, we have $\mathsf{lam}(x, \mathsf{lam}(z, K)) =_\alpha \mathsf{lam}(y, \mathsf{lam}(z', L))$. □

A.2. Proofs in Section 5

Proof of Proposition 18. By induction on X. We discuss only interesting cases, and omit easy proofs of 1, 3, 4 and 7–20.

Proof of 2. The case where X is $n\backslash Y$. We have $Y \in \mathsf{LD}_i$ by assumption $X \in \mathsf{LD}_i$. We also have $m \mid Y, n \mid Y$, and $m \perp n$ since $m \mid n\backslash Y$. By IH, we have $Y^j_m \in \mathsf{LD}_{i+1}$. We also have $n \mid Y^j_m$ by 1. Therefore, we have $n\backslash Y^j_m \in \mathsf{LD}_{i+1}$, that is, $(n\backslash Y)^j_m \in \mathsf{LD}_{i+1}$.

Proof of 5.

- The case X is $n\backslash Y$. By computing $(n\backslash Y)^j$, our goal becomes to show

  $$(1)\ Y^j \in \mathsf{LD}_i \quad \text{and} \quad (2)\ n \mid Y^j.$$

  We can show (1) by IH, and (2) by Proposition 18.4.
- The case X is k. By computing k^j, our goal becomes to show

  $$\begin{array}{ll} k \in \mathsf{LD}_i & \text{if } k < j, \\ \Box \in \mathsf{LD}_i & \text{if } k = j \end{array}$$

  and

  $$k - 1 \in \mathsf{LD}_i \quad \text{if } j < k$$

  from the assumptions $k < i + 1$ and $j < i + 1$; which is easily achieved.

Proof of 6.

- The case X is $n\backslash Y \in \mathsf{LD}_i$, that is, $Y \in \mathsf{LD}_i$ and $n \mid Y$. We have to show that $(n\backslash Y)_j \mid (n\backslash Y)^j$, namely, $Y_j \mid n\backslash Y^j$, which is equivalent to

  $$(1)\ Y_j \mid Y^j, \qquad (2)\ m \mid Y^j \quad \text{and} \quad (3)\ Y^j \perp n.$$

  We have (1) by IH. (2) follows from Proposition 18.4. (3) follows from Lemma 17.2.

- The case X is k, where we have to show $k_j \mid k^j$. By computing k_j and k^j, the goal becomes

$$\begin{array}{ll} 0 \mid k & \text{if } k < j, \\ 1 \mid \Box & \text{if } k = j \end{array}$$

and

$$0 \mid k - 1 \quad \text{if } k > j,$$

which is easily achieved. $\Box$

Proof of Proposition 19. By induction on X. We give outline of proofs of interesting cases here.
Proof of 1 *and* 2. We prove them simultaneously by showing that:

$$X \in \mathsf{LD}_i \Longrightarrow \langle X \rangle \in \mathsf{LD}_i \quad \text{and } \forall m.\ (m \mid X \Longrightarrow m \mid \langle X \rangle).$$

- The case where $X = n \backslash Y \in \mathsf{LD}_i$. In this case we have $n \mid Y$, $Y \in \mathsf{LD}_i$ and, by IH, $\langle Y \rangle \in \mathsf{LD}_i$ and $\forall m.\ (m \mid Y \Longrightarrow m \mid \langle Y \rangle)$, in particular, $n \mid \langle Y \rangle$. We have to show $\langle n \backslash Y \rangle \in \mathsf{LD}_i$ and $\forall m.\ (m \mid n \backslash Y \Longrightarrow m \mid \langle n \backslash Y \rangle)$, which are equivalent to:

$$(1)\ \langle Y \rangle^0_n \in \mathsf{LD}_{i+1} \quad \text{and} \quad (2)\ \forall m.\ (m \mid Y, n \mid Y, m \perp n \Longrightarrow m \mid \langle Y \rangle^0_n).$$

We can obtain (1) by applying Proposition 18.2, and (2) by applying Proposition 18.1. Note that the proof of (1) shows the necessity of proving Proposition 19.1 and 2 simultaneously.
- The case where $X = [Y] \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_{i+1}$ and, by IH, $\langle Y \rangle \in \mathsf{LD}_{i+1}$ and $\forall m.\ (m \mid Y \Longrightarrow m \mid \langle Y \rangle)$, We have to show $\langle [Y] \rangle \in \mathsf{LD}_i$ and $\forall m.\ (m \mid [Y] \Longrightarrow m \mid \langle [Y] \rangle)$, which are equivalent to:

$$\begin{array}{l} (1)\ \langle Y \rangle^0 \in \mathsf{LD}_i, \\ (2)\ \langle Y \rangle_0 \mid \langle Y \rangle^0 \end{array}$$

and

$$(3)\ \forall m.\ (m \mid Y \Longrightarrow m \mid \langle Y \rangle^0 \text{ and } \langle Y \rangle_0 \mid \langle Y \rangle^0 \text{ and } m \perp \langle Y \rangle_0).$$

We have (1) by Proposition 18.5, (2) by Proposition 18.6 and (3) by Proposition 18.{4, 6, 2}.

Proof of 3.

- The case where $X = n \backslash Y \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_i, n \mid Y$ and, by IH, $bd(\langle Y \rangle) = md(Y)$. We have to show $bd(\langle n \backslash Y \rangle) = md(n \backslash Y)$, which is equivalent to

$$bd(\langle Y \rangle^0_n) = md(Y).$$

We have this by Propositions 18.19, 19.2 and IH.
- The case where $X = [Y] \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_{i+1}$ and, by IH, $bd(\langle Y \rangle) = md(Y)$. We have to show $bd(\langle [Y] \rangle) = md([Y])$, which is equivalent to

$$bd(\langle Y \rangle^0) = md(Y).$$

We have this by Proposition 18.20 and IH.

Proof of 4 is similar to that of 3.
Proof of 5 can be obtained by using Proposition 19.{3, 4}.
Proof of 6.

- The case where $X = n\backslash Y \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_i$, $m \mid Y$, $n \mid Y$, $m \perp n$. We have to show $\langle m\backslash Y^j_n\rangle = \langle m\backslash Y\rangle^j_n$, or equivalently,

  $$\langle Y^j_m\rangle^0_n = (\langle Y\rangle^0_n)^{j+1}_m.$$

  We have this as follows.

  $$\begin{array}{lll} \langle Y^j_m\rangle^0_n &= (\langle Y\rangle^j_m)^0_n & \text{by IH} \\ &= (\langle Y\rangle^0_n)^{j+1}_m & \text{by Proposition 18.7, using Proposition 19.}\{1,2\}. \end{array}$$

- The case where $X = [Y] \in \mathsf{LD}_i$ and $m \mid [Y]$. In this case, we have $Y \in \mathsf{LD}_{i+1}$ and also $m \mid \langle [Y]\rangle$ by Proposition 19.2, and hence have $\langle Y\rangle_0 \perp m$ by Proposition 18.3. We have to show $\langle [Y]^j_m\rangle = \langle [Y]\rangle^j_m$, which is equivalent to

  $$(1)\ \langle Y^{j+1}_m\rangle_0 = \langle Y\rangle_0 \quad \text{and} \quad (2)\ \langle Y^{j+1}_m\rangle^0 = (\langle Y\rangle^0)^j_m.$$

  We have (1) by Proposition 18.8 using Proposition 19.$\{1,2\}$. We have (2) by Proposition 18.10 using Proposition 19.$\{1,2\}$.

  Proof of 7.

- The case where $X = n\backslash Y \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_i$ and $n \mid Y$. We have to show $\langle n\backslash Y\rangle_j = n\backslash Y_j$, or equivalently,

  $$(\langle Y\rangle^0_n)_{j+1} = Y_j.$$

  We have this as follows.

  $$\begin{array}{lll} (\langle Y\rangle^0_n)_{j+1} &= \langle Y\rangle_j & \text{by Proposition 18.8 using Proposition 19.}\{1,2\} \\ &= Y_j & \text{by IH.} \end{array}$$

- The case where $X = [Y] \in \mathsf{LD}_i$. In this case, we have $Y \in \mathsf{LD}_{i+1}$. We have to show $\langle [Y]\rangle_j = [Y]_j$, or equivalently,

  $$\langle Y\rangle^0_j = Y_{j+1}.$$

  We have this as follows.

  $$\begin{array}{lll} \langle Y\rangle^0_j &= \langle Y\rangle_{j+1} & \text{by Proposition 18.16 using Proposition 19.1.} \\ &= Y_{j+1}. & \text{by IH.} \end{array}$$

  Proof of 8.

- The case where $X = n\backslash Y \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_i$ and $n \mid Y$. We have to show $\langle (n\backslash Y)^j\rangle = \langle n\backslash Y\rangle^j$, or equivalently,

  $$\langle Y^j\rangle^0_n = (\langle Y\rangle^0_n)^{j+1}.$$

  We have this as follows.

  $$\begin{array}{lll} \langle Y^j\rangle^0_n &= (\langle Y\rangle^j)^0_n & \text{by IH} \\ &= (\langle Y\rangle^0_n)^{j+1} & \text{by Proposition 18.11 using Proposition 19.}\{1,2\}. \end{array}$$

- The case where $X = [Y] \in \mathsf{LD}_i$. In this case, we have $Y \in \mathsf{LD}_{i+1}$. We have to show $\langle [Y]^j\rangle = \langle [Y]\rangle^j$, or equivalently,

  $$(1)\ \langle Y^{j+1}\rangle_0 = \langle Y\rangle_0 \quad \text{and} \quad (2)\ \langle Y^{j+1}\rangle^0 = (\langle Y\rangle^0)^j.$$

  We have (1) as follows.

  $$\begin{array}{lll} \langle Y^{j+1}\rangle_0 &= (\langle Y\rangle^{j+1})_0 & \text{by IH} \\ &= \langle Y\rangle_0 & \text{by Proposition 18.17 using Proposition 19.1.} \end{array}$$

We have (2) as follows.

$$\begin{aligned}\langle Y^{j+1}\rangle^0 &= (\langle Y\rangle^{j+1})^0 && \text{by IH}\\ &= (\langle Y\rangle^0)^j && \text{by Proposition 18.18 using Proposition 19.1.}\end{aligned}$$

Proof of 9.

- The case where $X = n\backslash Y \in \mathsf{LD}_i$. In this case we have $Y \in \mathsf{LD}_i$ and $n \mid Y$. We have to show $\langle\langle n\backslash Y\rangle\rangle = n\backslash Y$, or equivalently,

  (1) $\langle\langle Y\rangle^0_n\rangle_0 = n$ and (2) $\langle\langle Y\rangle^0_n\rangle^0 = Y$.

  We have (1) as follows.

$$\begin{aligned}\langle\langle Y\rangle^0_n\rangle_0 &= (\langle\langle Y\rangle\rangle^0_n)_0 && \text{by Proposition 18.6 using Proposition 19.}\{1,2\}\\ &= (Y^0_n)_0 && \text{by IH}\\ &= n && \text{by Proposition 18.12.}\end{aligned}$$

  We have (2) as follows.

$$\begin{aligned}\langle\langle Y\rangle^0_n\rangle^0 &= (\langle\langle Y\rangle\rangle^0_n)^0 && \text{by Proposition 18.6 using Proposition 19.}\{1,2\}\\ &= (Y^0_n)^0 && \text{by IH}\\ &= Y && \text{by Proposition 18.13.}\end{aligned}$$

- The case where $X = [Y] \in \mathsf{LD}_i$. In this case, we have $Y \in \mathsf{LD}_{i+1}$. We have to show $\langle\langle [Y]\rangle\rangle = [Y]$, or equivalently

  $\langle\langle Y\rangle^0\rangle^0_{\langle Y\rangle_0} = Y$.

  We have this as follows

$$\begin{aligned}\langle\langle Y\rangle^0\rangle^0_{\langle Y\rangle_0} &= \langle\langle Y\rangle^0\rangle^0_{Y_0} && \text{by Proposition 18.7}\\ &= (\langle\langle Y\rangle\rangle^0)^0_{Y_0} && \text{by Proposition 18.8 using Proposition 19.1}\\ &= (Y^0)^0_{Y_0} && \text{by IH}\\ &= Y && \text{by Proposition 18.14.}\ \square\end{aligned}$$

References

[1] B. Aydemir, A. Charguéraud, B.C. Pierce, R. Pollack, S. Weirich, Engineering formal metatheory, in: Proceedings of the 35th Annual ACM Symposium on Principles on Programming Languages, POPL'08, 2008.

[2] H.P. Barendregt, The Lambda Calculus: Its Syntax and Semantics, revised ed., North-Holland, Amsterdam, 1984.

[3] N. Bourbaki, Elements of Mathematics: Theory of Sets, Addison-Wesley, 1968.

[4] A. Church, The Calculi of Lambda-Conversion, Prince University Press, 1941.

[5] N.G. de Bruijn, Lambda calculus notation with nameless dummies, a tool for automatic formal manipulation with application to the Church–Rosser theorem, Indag. Math. 34 (1972) 381–392.

[6] P. Dybjer, A general formulation of simultaneous inductive–recursive definitions in type theory, Journal of Symbolic Logic 65 (2000) 525–549.

[7] N. Forsberg, A. Setzer, Inductive–inductive definitions, in: Annual Conference on Computer Science Logic, CSL 2010, in: Lecture Notes in Computer Science, vol. 6247, Springer-Verlag, 2010.

[8] M.J. Gabbay, A. Pitts, A new approach to abstract syntax with variable binding, Formal Aspects of Computing 13 (2003) 341–363.

[9] D. Hendriks, V. van Oostrom, Adbmal, in: Proceedings of the 19th Conference on Automated Deduction, CADE 19, in: LNAI, vol. 2741, Springer-Verlag, 2003.

[10] S. Kahrs, Context rewriting, in: Conditional Term Rewriting Systems, CTRS'92, in: LNCS, vol. 656, Springer-Verlag, 1993.

[11] J. McKinna, R. Pollack, Some lambda calculus and type theory formalized, Journal of Automated Reasoning 23 (1999) 3–4.
[12] Y. Minamide, K. Okuma, Verifying CPS transformations in Isabelle/HOL, in: Proc. of the Second Workshop on Mechanized Reasoning about Languages with Variable Binding, 2003.
[13] A. Pitts, Nominal logic, a first order theory of names and binding, Information and Computation 186 (2003) 165–193.
[14] R. Pollack, M. Sato, W. Ricciotti, A canonical locally named representation of binding, Journal of Automated Reasoning (2011). http://dx.doi.org/10.1007/s10817-011-9229-y.
[15] W.V. Quine, Mathematical Logic, Norton, New York, 1940.
[16] M. Sato, Theory of symbolic expressions, I, Theoretical Computer Science 22 (1983) 19–55.
[17] M. Sato, An abstraction mechanism for symbolic expressions, in: V. Lifschitz (Ed.), Artificial Intelligence and Mathematical Theory of Computation (Papers in Honor of John McCarthy), Academic Press, 1991, pp. 381–391.
[18] M. Sato, M. Hagiya, Hyperlisp, in: Proceedings of the International Symposium on Algorithmic Language, North-Holland, 1981, pp. 251–269.
[19] M. Sato, R. Pollack, External and internal syntax of the lambda-calculus, Journal of Symbolic Computation 45 (2010) 598–616.
[20] M. Sato, T. Sakurai, Y. Kameyama, A. Igarashi, Calculi of meta-variables, in: M. Baaz, J.A. Makowsky (Eds.), Computer Science Logic, in: LNCS, vol. 2803, Springer, 2003.
[21] H. Schwichtenberg, Minlog, in: The Seventeen Provers of the World, in: LNAI, vol. 3600, Springer-Verlag, 2006.
[22] N. Shankar, A mechanical proof of the Church–Rosser theorem, Journal of the ACM 35 (3) (1988).
[23] C. Urban, Nominal techniques in isabelle/HOL, Journal of Automated Reasoning 40 (4) (2008).