Usability: formalising (un)definedness in typed lambda calculus

Jan Kuper

University of Twente, Department of Computer Science
P.O.Box 217, 7500 AE Enschede, The Netherlands

e-mail: jankuper@cs.utwente.nl

Abstract. In this paper we discuss *usability*, and propose to take that notion as a formalisation of (un)definedness in typed lambda calculus, especially in calculi based on PCF. We discuss some important properties that make usability attractive as a formalisation of (un)definedness. There is a remarkable difference between usability and solvability: in the untyped lambda calculus the solvable terms are precisely the terms with a head normal form, whereas in typed lambda calculus the usable terms are "between" the terms with a normal form and the terms with a (weak) head normal form.

1 Introduction

The elementary form of undefinedness arises on the level of natural numbers, when the evaluation of a (closed) term M of type **Nat** does not terminate, i.e., when M does not have a normal form. Such a term is also often called meaningless. However, for higher types it is not so evident which terms should be called meaningless. Analogous to the situation for ground types, it is often felt to be attractive to call a term M meaningless, or undefined, if M does not have a normal form. However, one of the desirable properties of meaningless terms is, that their identification may not lead to inconsistency. It is well known that in general terms without a normal form can *not* be identified consistently. For the untyped lambda calculus, this is shown in (Barendregt 1984, section 2.2). For typed calculi (with sufficient computing power) this is immediately seen: for example, the evaluation of infinite lists does not terminate, but clearly they may not be identified. The same holds for recursively defined functions. Hence, at second sight, it is not natural to consider terms without a normal form as meaningless.

As an alternative, Abramsky and Ong propose to take the terms with a weak head normal form as the meaningful ones (Abramsky 1990, Ong 1988). As an argument in favour of this proposal Abramsky and Ong mention that in lazy functional languages no evaluation takes place inside weak head normal forms. However, this is only true since values of function type are not acceptable as output values. If terms of function type would be acceptable as output values, then e.g. $\lambda x.1+1$ would be evaluated to $\lambda x.2$, i.e., there would be an evaluation step inside a whnf.

As a third alternative for representing meaningfulness we mention the notion of solvability, introduced in (Barendregt 1971). This notion is widely accepted as an adequate formalisation of meaningfulness in the untyped lambda calculus. However, it turns out that the standard definition of solvability is not adequate for typed lambda calculi (see section 3).

In this paper we introduce a generalisation of solvability, called usability, based on the following interpretation of meaningfulness:

> a term is meaningful if it can have a contribution to the outcome of a terminating computation.

The remaining part of this paper is organised as follows. In section 2 we define the calculus $\boldsymbol{\lambda}$, in sections 3 and 4 we introduce usability and compare it to solvability, in section 5 we compare usable terms with (head) normal forms (it turns out that in typed lambda calculus the usable terms are *not* precisely the terms with a head normal form), in section 6 we formulate the Genericity Lemma, and in section 7 we show that all unusable terms can be identified consistently.

2 The calculus

The calculus $\boldsymbol{\lambda}$ is an inessential variant of PCF (Plotkin 1977), i.e., it is a simply typed lambda calculus (i.e. $\rightarrow$ is the only type constructor) with two ground types: **Nat** and **Bool**. There are constants for the natural numbers ($\underline{0}$, $\underline{1}$, ...) and for the truth values (`true`, `false`).

There are also the following constants of function type: `Succ`, `Pred`, `Zero?`, $\mathtt{if}_\sigma$ with the obvious interpretations. Different from PCF, $\boldsymbol{\lambda}$ has a conditional $\mathtt{if}_\sigma$ of type $\mathbf{Bool}\rightarrow\sigma\rightarrow\sigma\rightarrow\sigma$ for each type σ. The types of the other constants are obvious.

Apart from variables ($x, y, \ldots$) and constants ($c, f, \ldots$), the calculus $\boldsymbol{\lambda}$ has the following terms (given that M, N are terms): MN, $\lambda x{:}\sigma.M$, $\mu x{:}\sigma.M$. The typing rules are standard, we only mention the rule for μ-terms:

$$\frac{x{:}\sigma \vdash M{:}\sigma}{\vdash (\mu x{:}\sigma.M) : \sigma}$$

Usually, we will leave out type information from terms, and write $\lambda x.M$, $\mu x.M$.

The reduction rules are also standard. The β- and μ-rule are

$$(\lambda x.M)N \rightarrow M[x{:=}N],$$
$$\mu x.M \quad \rightarrow M[x{:=}\mu x.M],$$

where $M[x{:=}N]$ denotes substitution. Some examples of the δ-rules are

$$\begin{aligned} \mathtt{Succ}\ \underline{n} \quad &\rightarrow \underline{n{+}1} \\ &\ \ \vdots \\ \mathtt{if}_\sigma\ \mathtt{true} \ &\rightarrow \mathbf{K}_\sigma, \\ \mathtt{if}_\sigma\ \mathtt{false} &\rightarrow \mathbf{K}^*_\sigma, \end{aligned}$$

where $\mathbf{K}$, $\mathbf{K}^*$ are $\lambda xy.x$ and $\lambda xy.y$ respectively.

We often write `if` L `then` M `else` N for `if` LMN. We also use standard abbreviations like $\lambda xy.M$ for $\lambda x.\lambda y.M$. We remark, that all δ-redexes are of the form fc, where f, c are constants. This implies that all constants of function type are strict in their first argument (this will be essential in definition 4).

We use standard notations such as $M \twoheadrightarrow N$, $M{=}N$, and $\boldsymbol{\lambda} \vdash M{=}N$. Clearly, $\boldsymbol{\lambda}$ has the same computing power as PCF, since $\mu x.M$ is equivalent to $\mathbf{Y}(\lambda x.M)$, where $\mathbf{Y}$ is a fixpoint combinator in PCF. Hence, all partial recursive functions are $\boldsymbol{\lambda}$-definable. Finally we mention that $\boldsymbol{\lambda}$ has the Church-Rosser property, and that the standardization theorem holds in $\boldsymbol{\lambda}$.

On several places below we will consider an extension of $\boldsymbol{\lambda}$ with product types. Then we will assume that there are terms of the form $\langle M_1, M_2\rangle$, constants $\boldsymbol{\pi_1}, \boldsymbol{\pi_2}$, and reduction rules $\boldsymbol{\pi_i}\langle M_1, M_2\rangle \to M_i$.

3 Solvability

In the untyped lambda calculus the notion of *solvability* is considered as an adequate formalisation of meaningfulness. Some reasons for this are that the Genericity Lemma (see section 6) holds for unsolvable terms, and that all unsolvable terms can be consistently identified. However, a direct generalisation of the standard definition of solvability towards $\boldsymbol{\lambda}$ does not work, since the above properties do not hold there. In this section we will give three different characterisations of solvability in the untyped lambda calculus, and show that the reformulation of these characterisations towards $\boldsymbol{\lambda}$ are *not* equivalent. In the next section we introduce the notion of usability and show that it is equivalent to the weakest variant of solvability.

Lemma 1 (Solvability). *Let $\lambda\mathbf{x}.M$ be a closure of M (i.e., $\mathbf{x}$ consists of the free variables of M). Then in the untyped lambda calculus the following are equivalent (notice that (a) is the standard definition of solvability):*

(a) $\exists \mathbf{N}\ (\lambda\mathbf{x}.M)\mathbf{N} = \mathbf{I}$,
(b) $\forall L\ \exists \mathbf{N}\ (\lambda\mathbf{x}.M)\mathbf{N} = L$,
(c) *there exists a normal form L such that* $\exists \mathbf{N}\ (\lambda\mathbf{x}.M)\mathbf{N} = L$.

Proof. $(a) \Rightarrow (b)$: $(\lambda\mathbf{x}.M)\mathbf{N} = \mathbf{I}$ implies $(\lambda\mathbf{x}.M)\mathbf{N}L = L$.

$(b) \Rightarrow (c)$: Immediate.

$(c) \Rightarrow (a)$: If L is in normal form, then L is in head normal form, hence L is solvable (cf. Barendregt 1984, 8.3.14). That is, there is a sequence $\mathbf{P}$ such that $(\lambda\mathbf{y}.L)\mathbf{P} = \mathbf{I}$, where $\lambda\mathbf{y}.L$ is a closure of L. Hence,

$$(\lambda\mathbf{y}.(\lambda\mathbf{x}.M)\mathbf{N})\mathbf{P} = \mathbf{I}.$$

Clearly, this can be brought into the form $(\lambda\mathbf{y}.\lambda\mathbf{x}.M)\mathbf{Q} = \mathbf{I}$ (if necessary, add $\mathbf{I}$'s to the right of $\mathbf{P}$), i.e., M is solvable. □

Based on this lemma we define three variants of solvability in $\boldsymbol{\lambda}$.

Definition 2 (Solvability in λ). Let $\lambda \mathbf{x}.M$ be a closure of M. Then M is

(a) *strongly solvable*, if there is a type σ such that

$$\exists \mathbf{N}\ (\lambda \mathbf{x}.M)\mathbf{N} = \mathbf{I}_\sigma,$$

(b) *medium solvable*, if there is a type σ such that for all terms L of type σ

$$\exists \mathbf{N}\ (\lambda \mathbf{x}.M)\mathbf{N} = L,$$

(c) *weakly solvable*, if there is (a type σ and) a term L (of type σ), L in normal form, such that

$$\exists \mathbf{N}\ (\lambda \mathbf{x}.M)\mathbf{N} = L. \quad \square$$

Clearly, in the definition of weak solvability, mentioning the type of L is superfluous. We give some examples of the various forms of solvability in λ.

Example 1.

1. *A variable x of type σ is strongly solvable: $\lambda x.x = \mathbf{I}_\sigma$.*
2. *The term $M \equiv \lambda x{:}\sigma'.\, \lambda y{:}\sigma' \to \sigma.\, yx$ is medium solvable, as can be seen as follows. Let L be any term of type σ. Then $Mx(\lambda x.L) = L$. In general M is not strongly solvable (take $\sigma \equiv \mathbf{Nat}$).*
3. *If M is a constant of ground type, then M is weakly solvable (immediate), but not medium solvable, or strongly solvable.*
4. *Ω is not weakly solvable.*

Clearly, the items (a), (b), (c) from definition 2 correspond to (a), (b), (c) from lemma 1. In the untyped lambda calculus we have (a) $\Leftrightarrow$ (b) $\Leftrightarrow$ (c), whereas, as can be seen from the examples above, in λ we only have (a) $\Rightarrow$ (b) $\Rightarrow$ (c).

Lemma 3. *In λ we have*

$$M \text{ is strongly solvable} \overset{(1)}{\Longrightarrow} M \text{ is medium solvable} \overset{(2)}{\Longrightarrow} M \text{ is weakly solvable}$$

Proof. "$\overset{(1)}{\Longrightarrow}$": As lemma 1 "($a$) $\Rightarrow$ (b)"; "$\overset{(2)}{\Longrightarrow}$": Immediate. $\square$

4 Strict contexts and usability

Based on the interpretation of meaningfulness, described in section 1, we will introduce *strict contexts*, and use this notion to define *usability*. Informally, a context $C[\_]$ is strict, if we can be sure that M is "used" in the (leftmost) evaluation of $C[M]$. If $C[M]$ has a normal form, we may conclude that M has a contribution to this normal form. In such a case we will call M *usable*.

The following definitions formalise this intuition.

Definition 4 (Strict Context). (*a*) In $\boldsymbol{\lambda}$ a *strict context* $C[\_]$ is inductively defined as follows:

- the empty context, $[\_]$, is strict
- if $C[\_]$ is a strict context, f a constant and M a term, then

$$(i)\ f(C[\_]),$$
$$(ii)\ (C[\_])M,$$
$$(iii)\ \lambda x.C[\_],$$
$$(iv)\ \mu x.C[\_].$$

are strict contexts.

(*b*) In the untyped lambda calculus the definition of *strict context* is obtained from part (*a*) by removing clauses (*i*) and (*iv*). □

We remark that part (*a*) of this definition remains unchanged if product types are added to $\boldsymbol{\lambda}$. That is to say, $\boldsymbol{\pi_i}(C[\_])$ is a strict context whenever $C[\_]$ is. However, $\langle C[\_], M\rangle$ and $\langle M, C[\_]\rangle$ are *not* strict contexts.

Lemma 5. *Let $C[\_]$ be a strict context.*

(*a*) *In $\boldsymbol{\lambda}$ a strict context is of one of the following five forms:*

$$(i)\ [\_],$$
$$(ii)\ f(C[\_])M_1\cdots M_n,\ n\geq 0,$$
$$(iii)\ (C[\_])M_1\cdots M_n,\quad n\geq 1, C[\_]\ \textit{not of form}\ (ii)\ \textit{or}\ (iii),$$
$$(iv)\ \lambda x_1\cdots x_n.C[\_],\quad n\geq 1, C[\_]\ \textit{not of form}\ \lambda x.C'[\_],$$
$$(v)\ \mu x_1\cdots x_n.C[\_],\quad n\geq 1, C[\_]\ \textit{not of form}\ \mu x.C'[\_].$$

(*b*) *In the untyped lambda calculus a strict context is of form* (*i*), (*iii*), *or* (*iv*).

Proof. By induction on the construction of $C[\_]$. □

The next definition works for any calculus in which strict contexts can be defined.

Definition 6 (Usability).

- A term M is *usable for computing* N, notation $M \gg N$, if there is a strict context $C[\_]$ such that $C[M] \twoheadrightarrow N$. We will sometimes call M *relatively usable*,
- M is *usable* if $M \gg N$ for some normal form N.

We call $\gg$ the *usability relation.* □

The notation $M \gg N$ was introduced by Barendregt in his Ph.D.-thesis (Barendregt 1971, definition 3.3.2), and pronounced as "N is in the solution of M". Barendregt defined $\gg$ for combinatory logic only. It did not show up in the literature again, since it was thought that it did not work for the lambda calculus.

We mention some examples.

Example 2.

1. *Normal forms are usable,*
2. *a variable $x{:}\sigma$ is usable to compute any term $M{:}\tau$, i.e., $x \gg M$ for every x, M and every σ, τ. As before, there is a term $\lambda y.\underline{0}$ of type σ (or* `true` *instead of $\underline{0}$). Since* $\mathtt{if}_\tau\ \mathtt{Zero?}((\lambda x.[\_])(\lambda y.\underline{0})\mathtt{y})\ \mathtt{then}\ M\ \mathtt{else}\ M$ *is a strict context, it follows that $x \gg M$.*
3. *$x\Omega$ is usable (see example 1),*
4. *Ω is not usable (see corollary 12),*
5. *if product types are added to $\boldsymbol{\lambda}$, then $\langle \underline{0}, \Omega \rangle$ is usable (since $\pi_1[\_]$ is a strict context), but $\langle \Omega, \Omega \rangle$ is not usable.*

In the next lemma we list some simple properties of (relative) usability.

Lemma 7.

(i)	$C[\_]$ *is strict*	$\Rightarrow$	$M \gg C[M]$,
(ii)	$M \twoheadrightarrow N$	$\Rightarrow$	$M \gg N$,
(iii)	$M \gg N, N \gg L$	$\Rightarrow$	$M \gg L$,
(iv)	$M \gg N$, N *is usable*	$\Rightarrow$	M *is usable,*
(v)	M	$\gg$	M,
(vi)	M	$\gg$	fM (f *a constant of function type),*
(vii)	M	$\gg$	MN,
$(viii)$	M	$\gg$	$\lambda x.M$,
(ix)	$\lambda x.M$	$\gg$	M,
(x)	M	$\gg$	$M[x{:=}N]$
(xi)	M *is usable*	$\Leftrightarrow$	$\lambda x.M$ *is usable,*
(xii)	$\boldsymbol{\lambda} \vdash M{=}N$	$\Rightarrow$	(M *usable* $\Leftrightarrow$ N *usable),*
$(xiii)$	M *is usable*	$\Leftrightarrow$	$M \gg c$ (c *a constant of ground type).*

Proof. Most of these properties are easy to prove, and left to the reader. With respect to (ix) we remark that $[\_]x$ is a strict context. Property (xi) can be proved using (iv), $(viii)$ and (ix).

For $(xiii)$ notice that there is a normal form N such that $M \gg N$. Hence N does not contain a μ-term. By (x) we may assume that N is closed. Now let $\mathbf{L}$ be a sequence of closed terms in normal form such that $N\mathbf{L}$ is of ground type. Since the μ-free fragment of $\boldsymbol{\lambda}$ is strongly normalising, it follows that $N\mathbf{L} \twoheadrightarrow c$ for some constant c. □

The following lemma makes explicit that usability is indeed a generalisation of solvability. If product types are added to $\boldsymbol{\lambda}$, then this lemma does not hold any more. For example, $\langle \underline{0}, \Omega \rangle$ is usable, but not (weakly) solvable (see example 2).

Lemma 8. (a) *In $\boldsymbol{\lambda}$: M is usable $\Leftrightarrow$ M is weakly solvable,*

(b) *In the untyped lambda calculus: M is usable $\Leftrightarrow$ M is solvable.* □

Proof. (a) "$\Leftarrow$": If M is weakly solvable, then there are sequences $\mathbf{x}, \mathbf{L}$ such that $(\lambda\mathbf{x}.M)\mathbf{L}$ has a normal form, N say. Since $(\lambda\mathbf{x}.[\_])\mathbf{L}$ is a strict context, it follows that $M \gg N$. Hence, M is usable.
(a) "$\Rightarrow$": Tedious. We have to prove that M is weakly solvable. It is sufficient to prove that there is a (strict) context $C[\_] \equiv (\lambda\mathbf{x}.[\_])\mathbf{L}$ such that $C[M]$ has a normal form.

If M is usable, then there is a strict context $C_0[\_]$ such that $C_0[M]$ has a normal form. Without loss of generality we may assume that $C_0[\_]$ is constructed without applying clause (iv) of definition 4: since $C_0[M]$ has a normal form, a "subcontext" of the form $\mu x.C'[\_]$ can be replaced by $(\lambda x.C'[\_])(\mu x.C'[M])$, which is also strict ($M$ is given).

Define a "measure function" q on strict contexts as follows:

$$\begin{aligned} q([\_]) &= 0 \\ q(fC[\_]) &= q(C[\_]) + 1 \\ q((C[\_])X) &= q(C[\_]) \\ q(\lambda x.C[\_]) &= \begin{cases} q(C[\_]) & \text{if } C[\_] \text{ is of the form } \lambda x.C'[\_] \\ q(C[\_]) + 1 & \text{otherwise} \end{cases} \end{aligned}$$

So q counts the number of applications of clause $4(i)$ and the number of sequences of consecutive "leading" lambda's. We proceed by induction on $q(C_0[\_])$.

Basic case: $q(C_0[\_]) = 0$. Then $C_0[\_] \equiv [\_]M_1 \cdots M_n$, $n \geq 0$, i.e., $C_0[\_]$ is of the required form already.

Induction case: $q(C_0[\_]) > 0$. Notice that $C_0[\_] \not\equiv [\_]$ (since $q([\_]) = 0$). Hence, by lemma 5, $C_0[\_]$ is of one of the following three forms:

1. $C_0[\_] \equiv \lambda x_1 \cdots x_n.C_1[\_]$, $n \geq 1$, $C_1[\_]$ not of the form $\lambda y.C_1'[\_]$. Since $C_0[M]$ has a normal form, $C_1[M]$ has a normal form too. Since

   $$q(C_1[\_]) < q(C_0[\_]),$$

   the result follows by the induction hypothesis.
2. $C_0[\_] \equiv f(C_1[\_])L_1 \cdots L_n$, $n \geq 0$. Since $C_0[M]$ has a normal form, it follows by that $C_1[M]$ has a normal form too. Since

   $$q(C_1[\_]) < q(C_0[\_]),$$

   the result follows by the induction hypothesis.
3. $C_0[\_] \equiv (C_1[\_])L_1 \cdots L_n$, $n \geq 1$, $C_1[\_]$ not of form (ii) or (iii) of lemma 5. Hence, $C_1[\_]$ is of one of the following two forms:
   (a) $C_1[\_] \equiv [\_]$. Then $C_0[\_]$ is of the required form.
   (b) $C_1[\_] \equiv \lambda x_1 \cdots x_m.C_2[\_]$, $m \geq 1$, $C_2[\_]$ not of form $\lambda x.C_2'[\_]$. Hence, by lemma 5, there are three possible forms for $C_2[\_]$, given below as i, ii, iii. Recapitulating,
   $$C_0[M] \equiv (\lambda\mathbf{x}.C_2[M])\mathbf{L}$$
   has a normal form. Without loss of generality, we may assume that the length of $\mathbf{L}$ is not smaller than the length of $\mathbf{x}$. This can be seen easily:

since $C_0[M]$ has a normal form, we may add terms in normal form to the right of $\mathbf{L}$, and the result will have a normal form again.

i. $C_2[\_] \equiv [\_]$. Then $C_0[\_]$ is of the required form.

ii. $C_2[\_] \equiv f(C_3[\_])N_1 \cdots N_k$, $k \geq 0$. Then
$C_0[M] \equiv (\lambda \mathbf{x}.f(C_3[M])\mathbf{N})\mathbf{L}$
Since the length of $\mathbf{L}$ is not smaller than the length of $\mathbf{x}$, it is easily seen that there are $\mathbf{P}, \mathbf{Q}$ such that
$C_0[M] = f((\lambda \mathbf{x}.C_3[M])\mathbf{P})\mathbf{Q}$.
Since $C_0[M]$ has a normal form, it follows that $(\lambda \mathbf{x}.C_3[M])\mathbf{P}$ has a normal form. Since
$q((\lambda \mathbf{x}.C_3[\_])\mathbf{P}) < q(C_0[\_])$
the result follows by the induction hypothesis.

iii. $C_2[\_] \equiv (C_3[\_])N_1 \cdots N_k$, $k \geq 1$. Then
$C_0[M] \equiv (\lambda \mathbf{x}.(C_3[M])\mathbf{N})\mathbf{L}$.
As before, there is a sequence $\mathbf{Q}$ such that
$C_0[M] = (\lambda \mathbf{x}.C_3[M])\mathbf{Q}$.
Clearly, $C_3[\_]$ is not of form (ii) or (iii) from lemma 5. So two cases remain:

A. $C_3[\_] \equiv [\_]$. Then $(\lambda \mathbf{x}.C_3[\_])\mathbf{Q}$ is of the required form.

B. $C_3[\_] \equiv \lambda \mathbf{z}.C_4[\_]$. Then
$q((\lambda \mathbf{x}.\lambda \mathbf{z}.C_4[\_])\mathbf{Q}) < q(C_0[\_])$
and the result follows by the induction hypothesis.

This completes the proof of (a).

(b) The proof of (b) is analogous. □

5 Syntactic characterization of usability

In the untyped lambda calculus the solvable terms are precisely the terms with a head normal form. However, in $\boldsymbol{\lambda}$ the usable terms can not be characterized in this way. Consider the following terms.

$M_1 \equiv \texttt{if Zero?(Pred } x\texttt{) then } \underline{0} \texttt{ else } \Omega,$

$M_2 \equiv \texttt{if Zero?(Succ } x\texttt{) then } \underline{0} \texttt{ else } \Omega.$

Clearly $(\lambda x.M_1)\underline{1} \twoheadrightarrow \underline{0}$, so M_1 is usable. On the other hand, M_2 is *not* usable since there is no constant $\underline{n}$ for which $\texttt{Zero?(Succ } \underline{n}\texttt{)} \twoheadrightarrow \texttt{true}$. However, both are in head normal form as defined below.

Definition 9 (Head normal form). Let H stand for head normal form (hnf). Then

$B ::= x \mid BM \mid fB$

$H ::= B \mid \lambda x.H \mid c$ □

Notice that the restriction of this definition to the untyped lambda calculus yields the standard definition of hnf. For a comparison with other definitions of hnf's in typed lambda calculi, cf. (Kuper 1994, chapter 6).

For the proof of the next lemma, see (Kuper 1994, lemma 6.2.6). Compare also (Barendregt 1984, section 8.3).

Lemma 10.

$$\begin{array}{lllll} (a) & \lambda x.M \textit{ has a hnf} & \Leftrightarrow & M \textit{ has a hnf} & \\ (b) & M[x := N] \textit{ has a hnf} & \Rightarrow & M \textit{ has a hnf} & \\ (c) & MN \textit{ has a hnf} & \Rightarrow & M \textit{ has a hnf} & \square \end{array}$$

Now we come to the main proposition of this section. Notice that from the examples above it follows that the converse arrows do not hold.

Proposition 11.

$$\begin{array}{lll} M \textit{ has a normal form} & \overset{(1)}{\Longrightarrow} & M \textit{ is usable} \\ & \overset{(2)}{\Longrightarrow} & M \textit{ has a head normal form.} \end{array}$$

Proof. "$\overset{(1)}{\Longrightarrow}$": If $M \twoheadrightarrow N$, N in normal form, then $M \gg N$, i.e., M is usable. "$\overset{(2)}{\Longrightarrow}$": If M is usable, then by lemma 8 M is weakly solvable, i.e., there are $\mathbf{x}, \mathbf{N}$ such that $(\lambda \mathbf{x}.M)\mathbf{N}$ has a normal form. By induction on the structure of terms it is easy to see that a normal form is also a head normal form, so $(\lambda \mathbf{x}.M)\mathbf{N}$ has a head normal form (by lemma 10). $\square$

If product types are added to $\boldsymbol{\lambda}$, then it depends on the precise definition of head normal form, whether "$\overset{(2)}{\Longrightarrow}$" of this proposition will still hold. For example, $\langle \underline{0}, \Omega \rangle$ is usable. However, this term is usually not considered as a head normal form, but as a weak head normal form.

We mention two corollaries of proposition 11.

Corollary 12. *Ω is not usable.*

Proof. Ω does not have a head normal form. $\square$

Corollary 13. *Let M be a closed term of ground type. Then M is usable iff M has a normal form.*

Proof. A closed term M of ground type is in normal form iff M is in head normal form, The proof is completed by proposition 11. $\square$

6 Genericity

In section 1 we called a term meaningful if it can have a contribution to a terminating computation. This conception of meaningfulness motivated the notion of usability. Now we make this conception of meaningfulness precise in a different way:

a term M is meaningful if there is a context $C[\_]$ such that (1) $C[M]$ has a normal form N, and (2) there is a term M' such that $C[M']$ does *not* evaluate to N.

The main result of this section is that both formalisations of meaningfulness are equivalent. An important lemma in proving this equivalence is the Genericity Lemma (lemma 30). This lemma is proved by generalising a technique from (Barendregt 1971). This proof differs strongly from the standard proof of the Genericity Lemma for the untyped lambda calculus cf. (Barendregt 1984,-proposition 14.3.24), where it is proved by a topological method.

Definition 14 (Generic). A term M is *generic*, if for all contexts $C[\_]$ we have:

$C[M]$ has an nf $\Rightarrow$ $\forall X\ C[X]$ has the same nf. □

We remark that the generic terms are the operationally least defined terms in the sense of (Plotkin 1977, Berry et al, 1985): a term M is *operationally less defined* than N, if

$C[M]$ has a normal form $\Rightarrow$ $C[N]$ has the same normal form.

Now we come to the main result of this section.

Theorem 15. *M is generic $\Leftrightarrow$ M is not usable.*

Proof. "$\Leftarrow$": By a corollary of the Genericity Lemma (corollary 31).
"$\Rightarrow$": By contraposition. Suppose M is usable. Then there is a strict context $C[\_]$ such that $C[M]$ has a normal form. Since Ω is not usable, $C[\Omega]$ does not have a normal form. Hence, M is not generic. □

In the remaining part of this section we prove the Genericity Lemma and some of its variants (see lemma 30 and its corollaries). In order to do so, we need an extension $\underline{\lambda}$ of λ. Informally, the terms of $\underline{\lambda}$ are the terms of λ in which subterms can be underlined, but no subterm is underlined more than once.

Definition 16 (Terms in $\underline{\lambda}$).

- If A is a λ-term, then A and $\underline{A}$ are $\underline{\lambda}$-terms,
- If A, B are $\underline{\lambda}$-terms, then AB, $\lambda x.A$ and $\mu x.A$ are $\underline{\lambda}$-terms.

Terms without underlinings (i.e., λ-terms) are called *line free.* □

The following operation removes underlinings from $\underline{\lambda}$-terms.

Definition 17 (Removal of underlinings).

- $|A| \equiv A$ if A is line free,
- $|\underline{A}| \equiv A$,
- $|AB| \equiv |A||B|$,
- $|\lambda x.A| \equiv \lambda x.|A|$,
- $|\mu x.A| \equiv \mu x.|A|$. □

Definition 18 (Substitution in $\underline{\lambda}$). In addition to the properties of substitution in λ, we have

$$\underline{A}[x := B] \equiv \underline{A[x := |B|]}. \quad \square$$

Definition 19 (Reduction in $\underline{\lambda}$).

(*i*) The β-, μ- and δ-rules are identical to the corresponding rules in λ, e.g., $(\lambda x.M)N \to M[x{:=}N]$, where M, N are $\underline{\lambda}$-terms,
(*ii*) If $A \to B$ in λ, then $\underline{A} \to \underline{B}$ in $\underline{\lambda}$,
(*iii*) There are four *underlining rules*:

$$\begin{aligned} \underline{A}B &\to \underline{A|B|}, \\ f\underline{A} &\to \underline{fA}, \\ \lambda x.\underline{A} &\to \underline{\lambda x.A}, \\ \mu x.\underline{A} &\to \underline{\mu x.A}. \quad \square \end{aligned}$$

Notation. One-step reduction in $\underline{\lambda}$ is denoted by $\underline{\to}$; as expected, $\underline{\twoheadrightarrow}$ is the reflexive and transitive closure of $\underline{\to}$.

The underlining rules of $\underline{\lambda}$ correspond to strict contexts as follows.

Lemma 20. *Let $C[\_]$ be a line free context. Then*

$$C[\_] \text{ is strict} \iff \forall X(C[\underline{X}] \underline{\twoheadrightarrow} \underline{C[X]}).$$

Proof. "$\Rightarrow$": Immediate.
"$\Leftarrow$": Clearly, $C[\underline{X}] \underline{\twoheadrightarrow} \underline{C[X]}$ by underlining rules only. The result follows by contraposition. $\square$

Lemma 21. *For $\underline{\lambda}$-terms A, B*

$$A \underline{\twoheadrightarrow} B \Rightarrow |A| \twoheadrightarrow |B|.$$

Proof. By induction on the length of the reduction $A \underline{\twoheadrightarrow} B$. $\square$

Notation. If M is a (proper) subterm of N, we will write $M \subseteq N$ $(M \subset N)$.

Lemma 22. *If $A \underline{\twoheadrightarrow} B$ and $\underline{B'} \subseteq B$, then there exists an $\underline{A'} \subseteq A$ such that $A' \gg B'$.*

Proof. By induction on the length of the reduction $A \underline{\twoheadrightarrow} B$.

Basic case: We have to check all one step reductions by a case analysis. The underlining rules (cf. definition 19) are easy: they follow immediately from lemma 7. The β-, μ-, δ-rules are tedious, but straightforward. For the details we refer the reader to (Kuper 1994, section 7.3).

Induction case: By the transitivity of $\gg$. $\square$

Lemma 23. *Let $A \twoheadrightarrow B$ in λ, and let A' be such that $|A'| \equiv A$. Then there exists a term B', $|B'| \equiv B$, such that $A' \underline{\twoheadrightarrow} B'$.*

Proof. By induction on the length of the reduction $A \twoheadrightarrow B$. It is easy to see that all one step reductions can be copied to $\underline{\lambda}$, if necessary after some intermediate applications of the underlining rules. The (straightforward, but tedious) check of all possibilities is left to the reader. □

Definition 24. We write $A \mathrel{\widehat{\equiv}} B$, if B can be obtained from A by replacing zero or more underlined subterms of A by other underlined subterms. □

For example, for all line free terms M, N we have $\underline{M} \mathrel{\widehat{\equiv}} \underline{N}$, and $\underline{ML} \mathrel{\widehat{\equiv}} \underline{NL}$.

Lemma 25. *The relation $\widehat{\equiv}$ is an equivalence relation.*

Proof. Straightforward. □

Lemma 26.

- *if M, N are λ-terms, then $M \widehat{\equiv} N \Leftrightarrow M \equiv N$,*
- *$\underline{M} \mathrel{\widehat{\equiv}} \underline{N}$,*
- *$MN \widehat{\equiv} L$ iff there are M', N' such that $L \equiv M'N'$, and $M \widehat{\equiv} M'$, $N \widehat{\equiv} N'$,*
- *$\lambda x.M \widehat{\equiv} L$ iff there is an M' such that $L \equiv \lambda x.M'$, and $M \widehat{\equiv} M'$,*
- *$\mu x.M \widehat{\equiv} L$ iff there is an M' such that $L \equiv \mu x.M'$, and $M \widehat{\equiv} M'$.*

Proof. Straightforward. □

Lemma 27. *If $M \widehat{\equiv} M'$ and $N \widehat{\equiv} N'$, then*

$$M[x := N] \mathrel{\widehat{\equiv}} M'[x := N'].$$

Proof. By induction on the structure of M. □

Lemma 28. *If $M \underline{\rightarrow} N$ and $M' \widehat{\equiv} M$, then there is an $N' \widehat{\equiv} N$ such that $M' \underline{\rightarrow} N'$.*

Proof. By induction on the length of the reduction $M \underline{\rightarrow} N$.

Basic case. Let X be the chosen redex in M. Notice that this implies that X is not of the form $\underline{X'}$. There are two possibilities:

1. There is a $\underline{P} \subseteq M$ such that $X \subseteq P$. Then clearly $M \widehat{\equiv} N$. Take $N' \equiv M'$, then $N' \widehat{\equiv} N$, and $M' \underline{\rightarrow} N'$.
2. There is no $\underline{P} \subseteq M$ such that $X \subseteq P$.
   Suppose $X \underline{\rightarrow} Y$, and let $M \equiv C[X]$. Then, $N \equiv C[Y]$. Clearly there exist $X', C'[\_]$ with $X' \widehat{\equiv} X$ and $C'[\_] \widehat{\equiv} C[\_]$, such that $M' \equiv C'[X']$.
   We have to construct $Y' \widehat{\equiv} Y$ such that $X' \underline{\rightarrow} Y'$, by considering all possible reduction rules by which $X \underline{\rightarrow} Y$ (details are left to the reader).

Induction case. By transitivity of $\underline{\rightarrow}$. □

Corollary 29. *Let N be line free, and suppose $M \underline{\rightarrow} N$. Then for all M' with $M' \widehat{\equiv} M$, we have $M' \underline{\rightarrow} N$.* □

Now we come to the Genericity Lemma.

Lemma 30 (Genericity Lemma). *Let M, N be λ-terms, M not usable, N a normal form. If $\lambda \vdash FM = N$, then for all X*

$$\lambda \vdash FX = N.$$

Proof. Since N is a normal form, it follows by the Church-Rosser property that $FM \twoheadrightarrow N$. Hence $F\underline{M} \twoheadrightarrow N'$ for some N' with $|N'| \equiv N$ (lemma 23).

Suppose $\underline{L} \subseteq N'$, then by lemma 22: $M \gg L$. Since N is in normal form, L is in normal form too, and so M is usable, which is a contradiction. Therefore, N' does not contain underlined subterms, i.e., $N' \equiv N$. Hence, by corollary 29, $F\underline{X} \twoheadrightarrow N$ for every term X. By lemma 21 it follows that $FX \twoheadrightarrow N$. □

Corollary 31. *Let M, N be $\boldsymbol{\lambda}$-terms, M not usable, N a normal form. If $\boldsymbol{\lambda} \vdash C[M]{=}N$, then for all X*

$$\boldsymbol{\lambda} \vdash C[X]{=}N.$$

Proof. Suppose $\mathbf{x}$ is a sequence of variables containing all variables that are free in M or X. Let y be a fresh variable. Then

$$\begin{aligned}(\lambda y.C[y\mathbf{x}])(\lambda \mathbf{x}.M) &= C[(\lambda \mathbf{x}.M)\mathbf{x}] \\ &= C[M] \\ &= N.\end{aligned}$$

M is not usable, hence $\lambda \mathbf{x}.M$ not usable (lemma 7). Hence, by the Genericity Lemma (lemma 30):

$$\begin{aligned}C[X] &= C[(\lambda \mathbf{x}.X)\mathbf{x}] \\ &= (\lambda y.C[y\mathbf{x}])(\lambda \mathbf{x}.X) \\ &= N. \quad \square\end{aligned}$$

7 Identification of unusable terms

In this section we prove that it is consistent to identify in $\boldsymbol{\lambda}$ all unusable terms (respecting their type, of course). Intuitively this means that all meaningless terms may be identified. We also prove that this identification is maximal in the sense that identifying a usable term to an unusable term (in addition to the identification of all unusable terms) is inconsistent.

Notation. The set of all equations $P{=}Q$ for which P, Q have the same type and P, Q are unusable, is denoted by $\mathcal{S}$.

Theorem 32. *$\boldsymbol{\lambda} + \mathcal{S}$ is consistent.*

Proof. By contraposition. Suppose $\boldsymbol{\lambda} + \mathcal{S}$ is inconsistent. We show that this implies that $\boldsymbol{\lambda}$ is inconsistent.

If $\boldsymbol{\lambda} + \mathcal{S}$ is inconsistent, then

$$\boldsymbol{\lambda} + \mathcal{S} \vdash \mathtt{true}{=}\mathtt{false}.$$

Suppose that in a proof of this there are n applications of equations from $\mathcal{S}$. Then this proof can be presented as follows:

$$\mathtt{true} = \cdots = C_1[P_1] = C_1[Q_1] = \cdots = C_n[P_n] = C_n[Q_n] = \cdots = \mathtt{false},$$

where $P_i = Q_i$, $i{=}1,\ldots,n$, are equations from $\mathcal{S}$. So, the displayed equalities $C_i[P_i] = C_i[Q_i]$ are proved by equations from $\mathcal{S}$, all other equalities are proved by the axioms of $\boldsymbol{\lambda}$.

Now proceed by induction on n. If $n{=}0$, it follows that none of the equations from $\mathcal{S}$ is used, i.e., $\boldsymbol{\lambda} \vdash \mathtt{true}{=}\mathtt{false}$, and we are done.

Let $n{>}0$. By the proof above we have

$$\boldsymbol{\lambda} \vdash \mathtt{true} = C_1[P_1].$$

Since P_1 is unusable, it follows by the Genericity Lemma (corollary 31) that

$$\boldsymbol{\lambda} \vdash \mathtt{true} = C_1[Q_1].$$

Since

$$\boldsymbol{\lambda} \vdash C_1[Q_1] = C_2[P_2],$$

it follows that

$$\boldsymbol{\lambda} \vdash \mathtt{true} = C_2[P_2],$$

i.e., $\mathtt{true}{=}\mathtt{false}$ is proved by $n{-}1$ applications of equations from $\mathcal{S}$. The result follows by the induction hypothesis. □

We prove the maximality of the set $\mathcal{S}$ in the sense as described above.

Theorem 33. *Let M be a usable term, P an unusable term, M and P have the same type. Then $\boldsymbol{\lambda} + \mathcal{S} + M{=}P$ is inconsistent.*

Proof. Consider the term

$$\mho \equiv \mu x.\ \mathtt{if}\ \mathtt{Zero?}\ x\ \mathtt{then}\ \underline{1}\ \mathtt{else}\ \underline{0},$$

which is of type **Nat**, and notice that $\boldsymbol{\lambda}{+}\mho{=}c$ is inconsistent for every constant c of type **Nat** (we remark that c is restricted to the *given* constants of $\boldsymbol{\lambda}$. Clearly, it would be possible to introduce, for example, a constant $\bot$, with rule $\Omega{\to}\bot$, but *not* $\bot{\to}\bot$, i.e., $\bot$ is a normal form. Then $\mho = \bot$ does not lead to inconsistencies). Clearly, for type **Bool** there is also such a term, which we will denote by $\mho$ too.

Hence, for every type σ there is a term

$$\mho_\sigma \equiv \lambda \mathbf{x}.\mho$$

of type σ. By lemma 7 it follows that $\mho_\sigma$ is not usable.

Suppose M, P are of type σ, then $P{=}\mho_\sigma \in \mathcal{S}$, and so

$$\boldsymbol{\lambda} + \mathcal{S} + M{=}P \ \vdash\ M{=}\mho_\sigma.$$

Since M is usable, it follows by lemma 8(a) that there are sequences $\mathbf{y}, \mathbf{N}$ such that $(\lambda \mathbf{y}.M)\mathbf{N}$ has a normal form. Without loss of generality we may assume that this term is closed. It follows, that there is a sequence $\mathbf{L}$ and a constant c of ground type such that $(\lambda \mathbf{y}.M)\mathbf{NL} = c$.

Hence, in the theory $\lambda + \mathcal{S} + M{=}P$ we can derive the following inconsistency:

$$\begin{aligned} c &= (\lambda \mathbf{y}.M)\mathbf{NL} \\ &= (\lambda \mathbf{y}.\mho_\sigma)\mathbf{NL} \\ &= (\lambda \mathbf{y}.\lambda \mathbf{x}.\mho)\mathbf{NL} \\ &= \mho. \end{aligned}$$

The final equality follows by reasoning on the types of the subterms. □

Acknowledgements

I thank Henk Barendregt and Maarten Fokkinga for the interesting and valuable discussions. I also thank one of the anonymous referees for his or her detailed corrections.

References

Abramsky, S. (1990), The Lazy Lambda Calculus, *in:* Turner, D.A. (Editor), *Research Topics in Functional Programming Languages*, Addison-Wesley, Reading, Massachusetts.

Barendregt, H.P. (1971), *Some extensional term models for combinatory logics and lambda calculi*, Ph.D. Thesis, Utrecht.

Barendregt, H.P. (1975), Solvability in lambda calculi, Colloque International de Logique, Clermont Ferrand, 209 – 219.

Barendregt, H.P. (1984), *The Lambda Calculus – Its Syntax and Semantics* (revised edition), North-Holland, Amsterdam.

Berry, G., P.-L. Curien, J.-J. Lévy (1985), Full abstraction for sequential languages: state of the art, *in:* Nivat, M. and J.C. Reynolds (Editors), *Algebraic Methods in Semantics*, Cambridge University Press, Cambridge, 89 – 132.

Kuper, J. (1994), *Partiality in Logic and Computation – Aspects of Undefinedness*, Ph.D.Thesis, Enschede.

Ong, C.-H.L. (1988), *The Lazy Lambda Calculus: an Investigation into the Foundations of Functional Programming*, Ph.D. Thesis, Imperial College, London.

Plotkin,G.D.(1977),LCF considered as a programming language, *Theoretical Computer Science* 5, 223 – 255.