Extending Partial Combinatory Algebras

I. Bethke, J.W. Klop, R. de Vrijer

Computing Science Institute/

CSI-R9920 October

Computing Science Institute Nijmegen
Faculty of Mathematics and Informatics
Catholic University of Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands

Extending Partial Combinatory Algebras
dedicated to Roger Hindley

Inge Bethke[1] and Jan Willem Klop[1,2,3] and Roel de Vrijer[3]

[1] *University of Nijmegen, Department of Computer Science*
[2] *CWI, Amsterdam*
[3] *Vrije Universiteit Amsterdam, Department of Computer Science*
inge@cs.vu.nl jwk@cwi.nl rdv@cs.vu.nl

We give a negative answer to the question whether every partial combinatory algebra can be completed. The explicit counterexample will be an intricately constructed term model, the construction and the proof that it works heavily depending on syntactic techniques. In particular, it is a nice example of reasoning with elementary diagrams and descendants. We also include a domain-theoretic proof of the existence of an incompletable partial combinatory algebra.

1. Introduction

Consider a structure $\mathfrak{A} = \langle A, s, k, \cdot \rangle$, where A is some set containing the distinguished elements s, k, equipped with a binary operation $\cdot$ on A, called application, which may be partial.

Notation 1.1

1 Instead of $a \cdot b$ we write ab; and in writing applicative expressions, the usual convention of association to the left is employed. So for elements $a, b, c \in A$, the expression $aba(ac)$ is short for $((a \cdot b) \cdot a) \cdot (a \cdot c)$.
2 $ab\downarrow$ will mean that ab is defined; $ab\uparrow$ means that ab is not defined. Obviously, an applicative expression can only be defined if all its subexpressions are.
3 If t_1, t_2 are applicative expressions, $t_1 \cong t_2$ means that either both $t_1\uparrow$ and $t_2\uparrow$, or $t_1\downarrow$ and $t_2\downarrow$ and $t_1 = t_2$.

Definition 1.2 A structure $\mathfrak{A}$ as indicated above is a *partial combinatory algebra* (pca) if for all $a, b, c \in A$:

1 $ka\downarrow$, $sa\downarrow$, $sab\downarrow$,
2 $kab \cong a$ and $sabc \cong ac(bc)$.

If moreover the application operator is total, that is, if $ab\downarrow$ for all $a, b \in A$, then $\mathfrak{A}$ is called *total*, or just a *combinatory algebra* (ca).

Note that instead of $kab \cong a$ we can equivalently write $kab = a$. In general, we have $t_1 = t_2$ whenever $t_1 \cong t_2$ and $t_2\downarrow$.

Examples 1.3

1 A well-known example of a pca is Kleene's $\mathfrak{K} = \langle \mathbb{N}, s, k, app \rangle$, where app is defined as the Kleene-bracket application from recursion theory: $app(m, n) \cong \{m\}(n)$, and s and k are appropriately chosen to satisfy the characterizing axioms. (See Example 5.6.5 in (Mitchell, 1996).) As a matter of fact, there are (infinitely) many possible choices for s, k. Each choice yields an alternative variant of $\mathfrak{K}$.

2 Another example are the Uniformly Reflexive Structures of (Strong, 1968) and (Wagner, 1969).

3 The 'initial' pca is obtained within Combinatory Logic (CL). Here one takes all closed, strongly normalizable CL-terms modulo convertibility by means of the S- and K-axiom. Application is defined iff the result is again strongly normalizing. In every pca all applicative expressions corresponding to strongly normalizing CL-terms are defined. In particular all normal forms are defined.

4 One might suppose that also the weakly normalizing CL-terms modulo CL-convertibility constitute a pca, with application defined iff the result is again weakly normalizing, analogous to the pca of SN-terms in (3). However this is not the case. For, consider $\omega = SII$ with $I = SKK$. Then $S \cdot K \cdot \omega \cdot \omega$ is defined, but $K \cdot \omega \cdot (\omega \cdot \omega)$ is undefined.

5 On the other hand, there is an interesting class of CL-terms that we will call PN, *persistently normalizable* CL-terms, for which the construction in (4) does yield a pca. Define $M \in PN$ iff every subterm of a reduct of M has a normal form. If SN is the set of strongly normalizable CL-terms and WN the set of weakly normalizable CL-terms, we have SN $\subseteq$ PN $\subseteq$ WN. An example of a term in PN - SN is:

$$[\mathrm{S(K(KI))(SII)}][\mathrm{S(K(KI))(SII)}].$$

6 (Asperti and Ciabatoni, 1995; Asperti and Ciabatoni, 1996) have introduced 'effective applicative structures' (eas); an eas is equivalent to a pca. See Remark 8.1 for a description of the notion eas.

Remark 1.4

1 In proper pca's (i.e. nontotal pca's) we have $sk \neq ki$ with $(i = skk)$. For, with $sk = ki$ it would follow from $skab \cong kb(ab)$ and $kiab \cong ib \cong b$ that every ab is defined.

2 Likewise, in every proper pca we have for all $a \in A$, that $k(ka) \neq s(k(ka))$. Otherwise it would follow from

$$k(ka)bc \cong kab \cong a$$

and

$$k(ka)bc = s(k(ka))bc \cong k(ka)b(bc) \cong ka(bc)$$

that $bc\downarrow$, for all $b, c \in A$.

The question we address in this paper is whether it will always be possible to extend a partial combinatory algebra to a total combinatory algebra by, if needed, supplementing the domain with new elements, and completing the application operation on the extended domain. Formally, the notion of extension is given by the following definition.

Definition 1.5 An *extension* of a partial combinatory algebra $\mathfrak{A} = \langle A, s, k, \cdot_A \rangle$ is a partial combinatory algebra $\mathfrak{B} = \langle B, s, k, \cdot_B \rangle$ (same s and k), such that $A \subseteq B$ and $\cdot_A = \cdot_B | dom(\cdot_A)$, i.e. $\cdot_A$ coincides with the restriction of $\cdot_B$ to $dom(\cdot_A)$.

So the question is whether every pca has a total extension. It was raised by H.P. Barendregt, G. Mitschke and D. Scott and included in the list of open problems at the Swansea lambda calculus meeting of September 1979, which was organized by Roger Hindley. We quote from (Hindley, 1980):

This question was originally asked about the time of the Swansea 1974 meeting, and it seems both important and difficult. Whatever the answer, some interesting mathematics will probably result.

The negative answer was announced in (Klop, 1982), in a short note. The present paper elaborates that announcement, along the lines of the sketch given there.

2. Heuristics

In (Hindley, 1980) it is already remarked that the following straightforward attempt to complete a pca $\mathfrak{A} = \langle A, s, k, \cdot \rangle$ fails: Add a new element $*$ to A, and extend $\cdot$ to $A \cup \{*\}$ by defining $ab = *$, if $ab \uparrow$ in $\mathfrak{A}$, and $*a = a* = ** = *$. The reason this does not work is that then for all $a \in A$, we have $a = ka* = *$.

Another reason why this fails is that in the pca of Example 1.3(3) (of the strongly normalizable CL-terms modulo convertibility) it is inconsistent to equate all not strongly normalizable terms. See e.g. (Barendregt, 1984).

Next, one might try to proceed by adding 'formal elements' $\underline{ab}$ whenever $ab \uparrow$, extending the application to such a and b by stipulating $ab = \underline{ab}$, and then dividing out the 'appropriate' equivalence relation. However, as our proof of the existence of incompletable pca's will show, no such procedure can be uniformly successful.

It is well-known that one can formulate a 'non-erasing' version CL_I of CL, using instead of S and K the basic combinators I, J, satisfying $Ia = a$, $Jabcd = ab(adc)$. Also, the well-known combinators B, C, I, W can be used for a non-erasing version of CL. It is obvious how to formulate the corresponding notion of 'non-erasing' pca, with distinguished elements i, j (or b, c, i, w). Now it is not hard to show that for such non-erasing pca's based on $\{i, j\}$ or $\{b, c, i, w\}$ with corresponding rules there is no problem in extending to a total ca along the lines mentioned above.

We will now describe the intuition behind our syntactic construction of a pca $\mathfrak{A}$ which cannot be completed. In completing a pca, a previously undefined expression kt_1t_2 may become equal to a previously defined expression t_1 by virtue of the k-equation

$$kt_1t_2 = t_1 \ .$$

Now suppose that a pca could be devised in such a way, that after any would-be completion we would be forced to have $kt_1t_2 = ks_1s_2$, where s_1 is again a previously defined expression, but such that in the original pca $t_1 \neq s_1$. Then the assumption that a completion exists would necessarily yield an inconsistency:

$$t_1 = kt_1t_2 = ks_1s_2 = s_1 \ .$$

A counterexample pca where this indeed happens can be realized as follows. Consider a pca $\mathfrak{A}$, containing distinct elements a, b, c, and such that:

$$s(sk)a = s(sk)b \ .$$

Then we have as follows a conversion between ac and bc:

$$\begin{aligned} s(sk)ac &= s(sk)bc \\ skc(ac) &= skc(bc) \\ k(ac)(c(ac)) &= k(bc)(c(bc)) \\ ac &= bc \ . \end{aligned}$$

Now suppose we can arrange that in $\mathfrak{A}$ we have $ac\downarrow$, $bc\downarrow$, $ac \neq bc$, but $c(ac)\uparrow$, $c(bc)\uparrow$. The above conversion between ac and bc will then be ruled out since it involves the undefined expressions $c(ac)$ and $c(bc)$. We will arrange that also other conversions between ac and bc will be ruled out; they 'essentially' amount to the one above and will contain subterms 'essentially' the same as the forbidden $c(ac)$ and $c(bc)$. Such a pca $\mathfrak{A}$ will be incompletable. For in any completion $\mathfrak{A}^*$ of $\mathfrak{A}$ the conversion between the distinct elements ac and bc would go through.

As a guideline for the construction of a pca $\mathfrak{A}$ as sketched above, one can think of a, b, c as 'bearers of undefinedness'. In small doses, isolated or in an application such as ac, bc they are harmless, but the presence in an expression of 'large' clusters of them, like $c(ac)$, where large means 'length $\geq$3', make the expression undefined. And also expressions which reduce, in the usual Combinatory Logic sense, to undefined expressions, will be undefined. (So e.g. $\omega(\omega a)$, where $\omega = sii$, is an undefined expression, since it reduces to the large cluster $aa(aa)$.)

The construction just sketched will be performed within Combinatory Logic. In order to construct elements a, b, c with the required properties, some new constants A, B and C will be added to the combinators S and K. Note, also in view of the completability of non-erasing pca's (see above), that the 'culprit' is the K-combinator, or, in the pca, the axiom for k, with its erasing effect.

In Figure 1 we have summarized the situation. The terms AC, BC are convertible using the axioms for S, K (the downward arrows) and the axiom $S(SK)AC = S(SK)BC$ (the horizontal transition). However this conversion is not valid as it leads through 'forbidden territory', namely through the area $\mathcal{U}$ of undefined expressions ($\mathcal{D}$ is the area of defined expressions). This area $\mathcal{U}$, the shaded, cone-like part of the figure, is an absolute barrier; every attempted conversion between the two terms AC, BC must pass the forbidden area. The forbidden area contains all expressions that reduce to the utmost forbidden terms on the bottom of the cone, namely those terms containing a 'large cluster', among them $C(AC)$ and $C(BC)$.

3. Combinatory Logic and traces in CL conversions

In this section we collect the necessary basic properties of Combinatory Logic reduction. We emphasize the use of elementary diagrams in constructing Church-Rosser diagrams, and introduce the notion of 'trace'.