J Log Lang Inf (2011) 20:161–168
DOI 10.1007/s10849-010-9124-7

Typability in Partial Applicative Structures

Inge Bethke · Piet Rodenburg

Published online: 19 March 2010

Abstract Adapting a claim of Kracht (Theor Comput Sci 354:131–141, 2006), we establish a characterization of the typable partial applicative structures.

Keywords Partial applicative structures · Typability

1 Introduction

If you apply the copula *is* to the adjective *dead* you get the predicate *is dead*; and if you apply the noun *Socrates* to this predicate, you get the sentence *Socrates is dead*. This is one way a grammarian could categorize words and groups of words and describe how they combine to form meaningful expressions. Now you could ask the opposite question: given certain combinations of words as meaningful, is there a grammatical categorization that explains these combinations? Kracht tried to answer this question (among other things) in Kracht (2006), for a simple, abstract, grammatical model. We give a description of the details of this model below, state and refute a variant of Kracht's theorem, and prove a less complex characterization along the same lines.

I. Bethke · P. Rodenburg (✉)
Section Theoretical Computer Science, Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands
e-mail: P.H.Rodenburg@uva.nl

I. Bethke
e-mail: I.Bethke@uva.nl

2 Preliminaries

A *partial applicative structure*, for short *pas*, consists of a set A and a partial binary operation $\cdot$ (*application*) on A. As is the custom in combinatory logic, we tend to omit the operation symbol $\cdot$, and suppress parentheses assuming association to the left, writing e.g. abc when we mean $(a \cdot b) \cdot c$. We write $ab\downarrow$ to express that ab exists, and $ab\uparrow$ to express that it doesn't. If we use ab in a *positive* statement, such as that it belongs to some set (in particular, $ab\uparrow$ is a negative), we imply that it exists.

The *polynomial operations* of a pas $\mathfrak{A}$ are the operations that can be constructed by composition and application from the projection and constant operations. That is,

1. $\lambda\vec{x}.x_i$ is a polynomial operation,
2. for every $a \in A$, $\lambda\vec{x}.a$ is a polynomial operation, and
3. if p_1 and p_2 are polynomial operations of the same arity, then $\lambda\vec{x}.p_1(\vec{x})p_2(\vec{x})$ is a polynomial operation.

The *trivial* polynomial operations are the ones that can be constructed without the application operation, i.e. without the use of clause 3.

A *congruence relation* of a pas $\mathfrak{A}$ is an equivalence relation θ of A that respects application in the sense that

$$a \equiv b \;\&\; c \equiv d \;\&\; ac\downarrow \;\&\; bd\downarrow \;\Rightarrow ac \equiv bd \;\; (\theta).$$

In particular, by $\omega_{\mathfrak{A}}$ we denote the congruence

$$\{\langle a, b\rangle \mid \text{ for all unary polynomial operations } p \text{ of } \mathfrak{A},\;\; p(a)\downarrow \Leftrightarrow p(b)\downarrow\}.$$

(It is written $\asymp_{\mathfrak{A}}$ in Kracht 2006.) We observe that, relative to the definition of congruence that we just gave, $\omega_{\mathfrak{A}}$ is the Leibniz congruence $\Omega_{\mathfrak{A}}(A)$ in the sense of Blok and Pigozzi (1987). The *quotient* of a pas $\mathfrak{A}$ over a congruence relation θ is $\mathfrak{A}/\theta = \langle A/\theta, \cdot\rangle$; its *congruence classes* are denoted by a/θ.

A *type system* is an absolutely free algebra $\mathfrak{T} = \langle T, \rightarrow\rangle$ with a single binary operation. The free generators of $\mathfrak{T}$ are the *ground types*—we make no assumptions about their cardinality, except that there must be at least one; the rest are *function types*. In our notation for function types we use association to the right: $\alpha \rightarrow \beta \rightarrow \gamma = \alpha \rightarrow (\beta \rightarrow \gamma)$. A subset X of T is *strict* if $\alpha \rightarrow \beta \in X$ only if $\alpha, \beta \in X$.

Definition 2.1 A $\mathfrak{T}$-*typed* pas is a quadruple $\mathfrak{A}_{\mathfrak{T}} = \langle \mathfrak{A}, \mathfrak{T}, S, \mapsto\rangle$ where $\mathfrak{A}$ is a pas, $\mathfrak{T}$ is a type system, S is a strict subset of T and $\mapsto$ is an injective assignment of subsets of A to the elements of S such that, for $a, b \in A$,

1. $ab\downarrow \Leftrightarrow \exists\alpha, \beta \in S\; (a \in A_{\alpha\rightarrow\beta} \;\&\; b \in A_\alpha \;\&\; ab \in A_\beta)$,
2. $\{A_\alpha \mid \alpha \in S\}$ is a partition of A.

If such an assignment exists for a pas $\mathfrak{A}$ and a type system $\mathfrak{T}$, we say $\mathfrak{A}$ is $\mathfrak{T}$-typable. Moreover, a pas is typable if for some type system $\mathfrak{T}$ it is $\mathfrak{T}$-typable. The type of an element a of a $\mathfrak{T}$-typed applicative structure $\mathfrak{A}$ is the unique $\alpha \in T$ such that $a \in A_\alpha$. The elements of S are the *inhabited* types.

In a typable pas, no element applies to itself: by 1., such an element a should have a function type $\alpha \to \beta$, and also the antecedent type α; so by 2. and injectivity of the assignment $\xi \mapsto A_\xi, \alpha \to \beta = \alpha$, contradicting the absolute freedom of the type system.

Example 2.2 We consider the *simply typed* λ*-calculus* $\Lambda^{\to}$ as introduced by Church. Its alphabet is that of lambda calculus enriched with type annotations taken from a type system $\mathfrak{O} = \langle O, \to \rangle$ with a single ground type 0. The well-typed terms are formed from typed variables by the formation rules

1. if t is a term of type $\alpha \to \beta$ and t' is a term of type α, then tt' is a term of type β, and
2. if t is a term of type β, then $\lambda x^\alpha.t$ is a term of type $\alpha \to \beta$.

We denote by $Ter(\Lambda^{\to})$ the set of well-typed lambda terms. $Ter(\Lambda^{\to})$ can be viewed as a pas by putting

$$t \cdot t' = \begin{cases} tt' & \text{if } tt' \text{ is well-typed, and} \\ \uparrow & \text{otherwise.} \end{cases}$$

Now $\langle Ter(\Lambda^{\to}), \mathfrak{O}, O, \mapsto \rangle$ is an $\mathfrak{O}$-typed pas with $\mapsto$ the trivial assignment which assigns to terms of type α the type α. If we restrict ourselves to closed terms, type 0 is uninhabited. Hence any strict $S \subseteq O$ that partitions the closed terms does not contain type 0 and in such a typing, $0 \to \alpha$ and $\alpha \to 0$ are—if inhabited—ground types. We will return to this issue in Example 4.3.

Lemma 2.3 *Let* $\mathfrak{A}$ *be a typed partial applicative structure. If* $a, b \in A$ *have the same type, then* $a \equiv b \ (\omega_{\mathfrak{A}})$.

Proof By induction on unary polynomials, show that $p(a)$ and $p(b)$ have the same type if either one exists. □

3 Towards a Characterization of Typability

In Theorem 10 of Kracht (2006), Kracht gives a characterization of typability for partial *combinatory* algebras, a proper subclass of partial applicative structures. By analogy, one may rephrase this characterization as follows:

3.1 *A pas* $\mathfrak{A}$ *is typable if and only if*

(Tarski's Principle) $a \equiv c \ (\omega_{\mathfrak{A}})$ *if and only if there exists a nontrivial unary polynomial operation* p *of* $\mathfrak{A}$ *such that* $p(a)\downarrow$ *and* $p(c)\downarrow$, *and*

(Well-foundedness) *for every* a *there exist* n *and* $b_0, \ldots, b_n$ *such that* $ab_0 \ldots b_n\uparrow$.

This is too simple. To see that this does not characterize typability for partial algebras in general, consider a pas with three elements a, b, c, and application specified by

	a	b	c
a	a	$\uparrow$	$\uparrow$
b	$\uparrow$	$\uparrow$	c
c	$\uparrow$	$\uparrow$	$\uparrow$

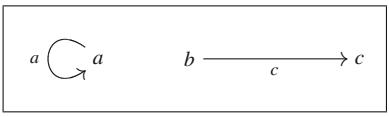

Table 1 Example 1

Table 1. Here an application matrix is given together with its graphical rendering. In general, such a graph should be read as follows: if $ab\downarrow$, then there is an arrow from a to b; and otherwise not. If the application ab results in c, that c (or if one wishes ab) is written as a label along the arrow. Labels can themselves be again the start of an arrow (see Example 2). Now, every element in Example 1 is in the domain of some nontrivial unary polynomial operation: $aa\downarrow$ and $bc\downarrow$, no nontrivial unary polynomial operation converges on more than one element, so ω is the diagonal relation; and the application is well-founded since ab, bb and cb all diverge. However, a cannot be given a type since it applies to itself.

This example indicates that the well-foundedness condition is too weak. We shall formulate a better condition below.

The first condition, however, is problematic as well, on two counts. First, $a \equiv c\ (\omega)$ if all nontrivial unary polynomials diverge on a and c. But, if the application operation of $\mathfrak{A}$ is void, $\mathfrak{A}$ is certainly typable. So the 'if and only if' in Tarski's Principle should be 'if'. And this will not be enough, for, second, consider the pas $\mathfrak{A}$ specified by Table 2. In Example 2 we have $a \not\equiv c\ (\omega_{\mathfrak{A}})$, for $abd\downarrow$ whereas $cbd\uparrow$. But ab and cb both

	a	b	c	ab	cb	d
a	$\uparrow$	ab	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
b	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
c	$\uparrow$	cb	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
ab	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	d
cb	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$
d	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

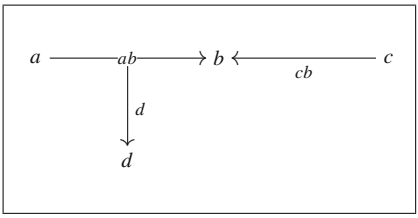

Table 2 Example 2

converge. So $\mathfrak{A}$ should not be typable. But here is a typing of the elements:

$$a : \beta \to (\delta \to \delta),\ b : \beta,\ c : \beta \to \gamma,\ ab : \delta \to \delta,\ cb : \gamma,\ d : \delta.$$

The problem is, that with 'nontrivial' we try to single out polynomial operations that really do something with their argument, but in xb, x does something rather than that something is done with it.

4 Characterization of Typability

We now give a charaterization of typability for partial applicative structures.

Definition 4.1 Define, for elements a, b of a pas $\mathfrak{A} = \langle A, \cdot \rangle$,

$$b <_{\mathfrak{A}} a \Leftrightarrow ab\downarrow \text{ or } \exists c \in A\ b = ac.$$

Observe that in the graphical rendering given in the examples we have $b < a$ if and only if $a \longrightarrow b$ or $a \underset{b}{\longrightarrow} c$ for some c. For finite structures, the nontypability then reveals itself by a cycle in the graph: the pas given in Example 2 is typable, Example 1 is not.

Theorem 4.2 (Main Theorem)
A pas $\mathfrak{A}$ is typable if and only if

1. *for all $a, c \in A$, if there exists $b \in A$ such that $ba\downarrow$ and $bc\downarrow$, then $a \equiv c\ (\omega_{\mathfrak{A}})$;*

2. *the relation $<_{\mathfrak{A}}$ is well-founded (in the usual sense).*

Proof ($\Rightarrow$) Assume a type system for $\mathfrak{A}$.

1. Suppose $ba\downarrow$ and $bc\downarrow$. Then there must be α, β such that $b \in A_{\alpha\to\beta}$ and $a, c \in A_\alpha$. Now since the types of a and c are the same, by Lemma 2.3, $a \equiv c\ (\omega_{\mathfrak{A}})$.

2. If $b <_{\mathfrak{A}} a$, the type of b is shorter than that of a.

($\Leftarrow$) Suppose $\mathfrak{A}$ satisfies 1. and 2. Let ω be $\omega_{\mathfrak{A}}$, $<$ be $<_{\mathfrak{A}}$. Define:

$$S_0 = \{\alpha \in A/\omega \mid \exists a \in \alpha\ \forall b \in A\ ab\uparrow\}.$$

We let S_0 be the collection of ground types, and for $\alpha \in S_0$ put $A_\alpha = \alpha$. Function types are defined by

$$A_{\alpha\to\beta} = \{a \in A \mid \exists b \in A_\alpha\ ab \in A_\beta\}.$$

Let a be a minimal element of $A - \bigcup_\alpha A_\alpha$. Then $a \notin \bigcup S_0$, so $ab\downarrow$ for some b. Since $b, ab < a$, there are α, β such that $b \in A_\alpha$ and $ab \in A_\beta$. Then $a \in A_{\alpha\to\beta}$. So every element has a type.

Now we prove by simultaneous induction on $<$:

- if $a \in A_\alpha \cap A_\beta$, then $\alpha = \beta$;

- if $a \in A_\alpha$, then $a/\omega \subseteq A_\alpha$.

Let a be minimal among the elements that do not satisfy these conditions. Suppose $a \in \alpha \in S_0$. If $a \in A_\beta$, with $\alpha \neq \beta$, then $\beta \notin S_0$, for then α and β would be distinct ω-congruence classes, and hence disjoint. So β is a function type; say $ab\downarrow$. Let $c \in A_\alpha$ be such that $cd\uparrow$ for all $d \in A$. Since $p(a)\downarrow$, for $p(x) = xb$, and $c \equiv a\ (\omega)$, we have

$cb = p(c)\downarrow$, quod non. The definition of the ground types ensures that a satisfies the second condition.

Now suppose $a \in A_{\alpha\to\beta} \cap A_{\gamma\to\delta}$. Then there are $b \in A_\alpha$ and $c \in A_\gamma$ such that $ab \in A_\beta$ and $ac \in A_\delta$. Hence by condition 1., $b \equiv c\ (\omega)$. So by induction hypothesis, $\alpha = \gamma$. Since ω is a congruence relation, a fortiori $ab \equiv ac\ (\omega)$, so $\beta = \delta$. Hence $\alpha \to \beta = \gamma \to \delta$.

Finally, suppose $a \in A_{\alpha\to\beta}$ and $b \in a/\omega$. Then for some $c \in A_\alpha$, $ac \in A_\beta$. Then for $p(x) = xc$, $p(a)\downarrow$, hence $p(b)\downarrow$, i.e. $bc\downarrow$. So $b \in A_{\alpha\to\gamma}$, with $bc \in A_\gamma$. Since ω is a congruence relation, and $a \equiv b\ (\omega)$, we have $ac \equiv bc\ (\omega)$. Since $ac < a$, by induction hypothesis $\beta = \gamma$. So $\alpha \to \beta = \alpha \to \gamma$. □

Example 4.3 We consider again $\Lambda^\to$ restricted to closed well-typed terms. If for two terms t and t' there exists a term t'' such that $t''t\downarrow$ and $t''t'\downarrow$, then they must have the same type, and hence $t \equiv t'\ (\omega)$. Thus condition 1. is satisfied. To see that also condition 2. holds, note that if $t < t'$ then for some $\alpha, \beta \in O$, t' has type $\alpha \to \beta$ and t is either of type α or type β. It now follows from the Main Theorem that the closed terms are typable. The proof suggests the following typing.

Let $\mathfrak{C} = \langle C, \to\rangle$ be the subsystem of $\mathfrak{O}$ consisting of the types that contain closed terms. If $tt'\uparrow$ for all closed terms t' then there are $\alpha, \beta \in O$ such that t is of type $\alpha \to \beta$ and no closed term has type α. We have therefore the set of ground types

$$S_0 = \{\alpha \to \beta \mid \alpha \to \beta \text{ is inhabited and } \alpha \text{ is not inhabited}\}$$

and for $\alpha \in S_0$ put $A_\alpha = \{t \mid t \text{ is a closed term of type } \alpha\}$. Function types are

$$A_{\alpha\to\beta} = \{t \mid \text{t is a closed term and } tt' \in A_\beta \text{ for some } t' \in A_\alpha\}.$$

The proof of the Main Theorem shows that this is indeed a type system for the closed simply typed lambda terms. For example, if $t' = \lambda x^0.x^0$ then $t' \in A_\alpha$ where $\alpha = 0 \to 0$. Moreover, since $\beta = (0 \to 0) \to 0$ is not inhabited and

$$t'' = (\lambda x^\alpha y^\beta.y^\beta x^\alpha)t'$$

is a closed term of type $\gamma = ((0 \to 0) \to 0) \to 0$ it follows that $t'' \in A_\gamma$. Hence

$$\lambda x^\alpha y^\beta.y^\beta x^\alpha \in A_{\alpha\to\gamma}.$$

The proof of the Main Theorem does not seem to leave much freedom in the choice of a type structure. We shall further investigate this point. Let us call an element x of a pas $\mathfrak{A}$ *isolated* if there is no $y \in A$ such that either xy or yx exists. A pas is *connected* if it has no isolated elements.

Theorem 4.4 *Let $\mathfrak{A}$ be a connected pas. Let $\mathfrak{A}_\mathfrak{T}$ and $\mathfrak{A}_\mathfrak{V}$ be expansions of $\mathfrak{A}$ to, respectively, a $\mathfrak{T}$-typed and a $\mathfrak{V}$-typed applicative structure. Then the relative subalgebras of inhabited types of $\mathfrak{T}$ and $\mathfrak{V}$ are isomorphic.*

Proof Let S be the set of inhabited types of $\mathfrak{A}_{\mathfrak{T}}$, and U that of $\mathfrak{A}_{\mathfrak{V}}$. Let $\alpha \mapsto A_\alpha$ be the type assignment of $\mathfrak{A}_{\mathfrak{T}}$, and $\sigma \mapsto B_\sigma$ that of $\mathfrak{A}_{\mathfrak{V}}$. Consider the definition

$$(\dagger) \quad \phi(\alpha) = \sigma \Leftrightarrow A_\alpha \cap B_\sigma \neq \emptyset.$$

We show simultaneously by induction on α that (†) implies

$$(\ddagger) \quad \text{for all } \sigma \in V,\ \phi(\alpha) = \sigma \Rightarrow A_\alpha = B_\sigma,$$

which makes (†) a proper definition of a mapping from S to U, and that ϕ is a homomorphism from the relative subalgebra $\mathfrak{S}$ of $\mathfrak{T}$ to the relative subalgebra $\mathfrak{U}$ of $\mathfrak{V}$.

If α is a ground type, and $a \in A_\alpha \cap B_\sigma$, then there is some $x \in A$ such that $xa\!\downarrow$. Then there must be $\beta \in S$ and $\tau \in U$ such that $x \in A_{\alpha\to\beta} \cap B_{\sigma\to\tau}$. Then $xy\!\downarrow$ for every $y \in B_\sigma$, so $B_\sigma \subseteq A_\alpha$, and likewise $A_\alpha \subseteq B_\sigma$.

If α is a function type, say $\alpha = \beta \to \gamma$, and $a \in A_\alpha \cap B_\sigma$, then $ax\!\downarrow$ for some $x \in A_\beta$. (Since S is strict, A_β must be inhabited.) Then σ must be a function type as well, say $\sigma = \tau \to \upsilon$. Since $x \in A_\beta \cap B_\tau$, and $ax \in A_\gamma \cap B_\upsilon$, we have $\phi(\beta) = \tau$, $\phi(\gamma) = \upsilon$, hence $\phi(\alpha) = \phi(\beta) \to \phi(\gamma)$, and $A_\beta = B_\tau$ and $A_\gamma = B_\upsilon$. Now $z \in A_{\beta\to\gamma}$ if and only if for all $x \in A_\beta$, $zx \in A_\gamma$, and analogously for $z \in A_{\tau\to\upsilon}$. So $A_\alpha = B_\sigma$.

By (‡) it is immediate that ϕ is injective. Surjectivity follows from (†) since $\{A_\alpha \mid \alpha \in S\}$ and $\{B_\sigma \mid \sigma \in U\}$ are partitions of A. □

It is clear from the proof that the type system would still be unique if $\mathfrak{A}$ had a single isolated element, since the construction for the 'if'-direction of the Main Theorem puts all the isolated elements into a single type.

Example 4.5 Since there exist no isolated (closed or open) simply well-typed lambda terms, we may conclude that the typing in both cases is unique up to isomorphism.

5 Related Work

Kracht's Theorem 10 is 3.1 with 'partial applicative structure' replaced by 'partial combinatory algebra', if we read 'typed combinatory algebra' as 'typed applicative system', which seems warranted by Kracht's description. We assume that partial combinatory algebras are defined as in Bethke et al. (1999); the distinguished elements s and k may be hidden. Then

- Partial combinatory algebras are not typable. For, by definition in such an algebra $kk\!\downarrow$; so k cannot be typed.

- Partial combinatory algebras that satisfy Tarski's Principle are total. To wit, since for any a, $ka\!\downarrow$, $a \equiv k\ (\omega_{\mathfrak{A}})$ by the Principle; so $kb\!\downarrow$ implies $ab\!\downarrow$.

So the theorem is false.

One of the referees brought Newman's remarkable paper (Newman 1943; Hindley 2008) to our attention, in which the author considers stratification and typing for various formalisms. Newman develops an idiosyncratic conceptual framework which

makes it difficult to compare his results to ours, but the relation seems to be as follows. On the one hand, Newman deals only with absolutely free algebras, and in this sense our work is more general. On the other hand, however, Newman considers algebras of arbitrary type, not just applicative structures. What the precise relation is, and how our work is to be generalized to encompass all of Newman's results, is a subject for further investigation.

Acknowledgments The authors are indebted to Henk Barendregt, Jan Willem Klop, Roel de Vrijer and an anonymous referee for valuable remarks and suggestions.

References

Bethke, I., Klop, J. W., & De Vrijer, R. (1999). Extending partial combinatory algebras. *Mathematical Structures in Computer Science, 9*, 483–505.

Blok, W. J., & Pigozzi, Don (1987). Algebraizable logics. *Memoirs of the AMS 396*. Providence, Rhode Island, USA: American Mathematical Society.

Hindley, J. R. (2008). M.H. Newman's typability algorithm for lambda-calculus. *Journal of Logic and Computation, 18*(2), 229–238.

Kracht, M. (2006). Partial algebras, meaning categories and algebraization. *Theoretical Computer Science, 354*, 131–141.

Newman, M. H. A. (1943). Stratified systems of logic. *Proceedings of the Cambridge Philosophical Society, 39*, 69–83.