Partial Combinatory Algebras

Andrej Bauer

January 5, 2025

Abstract

We outline a plan of formalization for partial combinatory algebras (PCA), including combinatory completeness, programming with PCAs, and some examples of PCAs. Time permitting, we will formalize the typed version as well.

This blueprint, and more generally the `partial-combinatory-algebras` Lean project, is part of the teaching materials for the course Formalized mathematics and proof assistants. It serves as a model blueprint to help students create their own class project blueprints.

1 Basic definitions

The set of *partial elements* $\widetilde{A}$ in A is

$$\widetilde{A} = \{u \subseteq A \mid \forall x, y \in u.\, x = y\}.$$

We think of $\emptyset$ as the undefined value and $\{a\}$ the defined, or *total* value $a \in A$. We construe each $a \in A$ also as the element $\{a\} \in \widetilde{A}$, i.e., there is an implicit coercion $A \to \widetilde{A}$.

Given $u \in \widetilde{A}$, write $u \Downarrow$ to mean that u is defined, i.e., $\exists a \in A.\, a \in u$.

Initial experimentation with formalization shows that it is advantageous to work with partial elements rather than elements, as that avoids coercions and helps the rpoof assistant compute types. When a partial element should be total this is expressed as a separate claim.

Definition 1.1. A *partial application* on a set A is a map $- \cdot_A - : \widetilde{A} \times \widetilde{A} \to \widetilde{A}$.

We also write $u\,v$ and $u \cdot v$ in place of $u \cdot_A v$ and we associate application to the left, so that $u \cdot v \cdot w = (u \cdot v) \cdot w$.

The preceding defintion is non-standard, as one usually expects $- \cdot_A - : A \times A \to \widetilde{A}$. However, as explained above, we prefer our definition as it is easier to work with. You might also have expected strictness conditions stating that $u \cdot_A v \Downarrow$ implies $u \Downarrow$ and $v \Downarrow$, which we did have in the initial formalization, but they were never used. It looks like our formalization already lead to a small discovery, namely that partial combinatory algebras work just as well when its elements can be applied to undefined values.

Definition 1.2. A set A with a partial map $- \cdot_A -$ is a *partial combinatory algebra (PCA)* if

there are elements $\mathsf{K}, \mathsf{S} \in \widetilde{A}$ such that, for all $u, v, w \in \widetilde{A}$:

$$
\begin{aligned}
&K\Downarrow, && S\Downarrow,\\
&u\Downarrow \Rightarrow \mathsf{K}\,u\Downarrow, && u\Downarrow \Rightarrow \mathsf{S}\,u\Downarrow,\\
&u\Downarrow \wedge v\Downarrow \Rightarrow \mathsf{K}\,u\,v\Downarrow, && u\Downarrow \wedge v\Downarrow \Rightarrow \mathsf{S}\,u\,v\Downarrow,\\
&u\Downarrow \wedge v\Downarrow \Rightarrow \mathsf{K}\,u\,v = u, && u\Downarrow \wedge v\Downarrow \wedge w\Downarrow \Rightarrow \mathsf{S}\,u\,v\,w\Downarrow,\\
& && u\Downarrow \wedge v\Downarrow \wedge w\Downarrow \Rightarrow \mathsf{S}\,u\,v\,w = (u\,w)\,(v\,w),
\end{aligned}
$$

2 Combinatory completeness

Definition 2.1. Given a set Γ of *variables* and a set A of elements, the *expressions* $\mathsf{Expr}\,\Gamma\,A$ are generated inductively by the following clauses:

- the constants `K` and `S` are expressions,
- an element $a \in A$ is an expression,
- a variable $x \in \Gamma$ is an expression,
- if e_1 and e_2 are expressions then $e_1 \cdot e_2$ is an expression.

The application $e_1 \cdot e_2$ in the third clause of the definition is formal, i.e., it is just a constructor of an inductive datatype. We again deviated from the standard definition in an inessential way by including primitive constants K and S, separate from the corresponding elements of A.

Given a set of variables Γ and a set A, a *valuation* is a map $\eta : \Gamma \to A$ which assigns elements to variables.

Definition 2.2. Given a valuation $\eta : \Gamma \to A$, $x \in A$ and $a \in a$, we may *override* η to get a valuation $\eta[x \mapsto a] : \Gamma \to A$ such that

$$(\eta[x \mapsto a])y = \begin{cases} \eta(y) & \text{if } y \in \Gamma,\\ a & \text{if } y = x. \end{cases}$$

Definition 2.3. The *evaluation* $[\![e]\!]\,\eta$ of an expression $e \in \mathsf{Expr}\,\Gamma\,A$ in a PCA A at a valuation $\eta : \Gamma \to A$ is defined recursively by the clauses

$$
\begin{aligned}
[\![\mathsf{K}]\!]\,\eta &= \mathsf{K}\\
[\![\mathsf{S}]\!]\,\eta &= \mathsf{K}\\
[\![a]\!]\,\eta &= a && \text{if } a \in A\\
[\![x]\!]\,\eta &= \eta(x) && \text{if } x \in \Gamma\\
[\![e_1 \cdot e_2]\!]\,\eta &= ([\![e_1]\!]\,\eta) \cdot_A ([\![e_2]\!]\,\eta).
\end{aligned}
$$

Definition 2.4. An expression $e \in \mathsf{Expr}\,\Gamma\,A$ is *defined* when $[\![e]\!]\,\eta\Downarrow$ for all $\eta : \Gamma \to A$.

Definition 2.5. The *abstraction* of an expression $e \in \mathsf{Expr}\,\Gamma\,A$ with respect to a variable $x \in \Gamma$ is the expression $\langle x\rangle\,e \in \mathsf{Expr}\,\Gamma\,A$ defined recursively by

$$
\begin{aligned}
\langle x\rangle\,x &= \mathsf{S}\cdot\mathsf{K}\cdot\mathsf{K},\\
\langle x\rangle\,y &= \mathsf{K}\cdot y && \text{if } x \neq y \in \Gamma,\\
\langle x\rangle\,a &= \mathsf{K}\cdot a && \text{if } a \in A,\\
\langle x\rangle\,(e_1 \cdot e_2) &= \mathsf{S}\cdot(\langle x\rangle\,e_1)\cdot(\langle x\rangle\,e_2).
\end{aligned}
$$

Proposition 2.6. *An abstraction $\langle x \rangle\, e$ is defined.*

Proof. By straightforward structural induction on e. □

Proposition 2.7. *Let $x \in \Gamma$ and $e \in \mathsf{Expr}\,(\Gamma \cup \{x\})\,A$. Then for every $a \in A$ and $\eta : \Gamma \to A$*

$$[\![(\langle x \rangle\, e) \cdot a]\!]\,\eta = [\![e]\!]\,\eta[x \mapsto a].$$

Proof. By straightforward structural induction on e. □

3 Programming with PCAs

We next show how to write programs in a PCA A. We call the elements of A accomplishing various programming tasks *combinators*.

Definition 3.1 (Identity)**.** There is the *identity combinator* $\mathsf{I} \in A$ such that $I\,a = a$ for all $a \in A$.

Definition 3.2 (Pairing)**.** There are combinators $\mathsf{pair}, \mathsf{fst}, \mathsf{snd} \in A$ such that, for all $a, b \in A$,

$$\mathsf{fst}\,(\mathsf{pair}\,a\,b) = a, \qquad\qquad \mathsf{snd}\,(\mathsf{pair}\,a\,b) = b.$$

Definition 3.3 (Booleans)**.** There are combinators $\mathsf{ite}\,\mathsf{fal}, \mathsf{tru} \in A$ such that, for all $a, b \in a$,

$$\mathsf{ite}\,\mathsf{tru}\;, a\,b = a \text{and} \mathsf{ite}\,\mathsf{fal}\,a\,b = b.$$

Definition 3.4 (Numerals)**.** For each $n \in \mathbb{N}$ there is $\overline{n} \in A$, as well as combinators $\mathsf{succ}, \mathsf{primrec} \in A$ such that, for all $n \in \mathbb{N}$ and $a, f \in A$,

$$\begin{aligned} \mathsf{succ}\,\overline{n} &= \overline{n+1}, \\ \mathsf{primrec}\,a\,f\,\overline{0} &= a, \\ \mathsf{primrec}\,a\,f\,\overline{n+1} &= f\,\overline{n}\,(\mathsf{primrec}\,a\,f\,\overline{n}). \end{aligned}$$

Definition 3.5. There is a combinator $\mathsf{Y} \in A$ such that, for all $a \in A$,

$$\mathsf{Y}\,a = a(\mathsf{Z}\,a).$$

Definition 3.6 (General recursion)**.** There is a combinator $\mathsf{Z} \in A$ such that, for all $f, a \in A$,

$$\mathsf{Z}\,f \Downarrow, \qquad \text{and} \qquad \mathsf{Z}\,f\,a = f(\mathsf{Z}\,f)\,a.$$

Definition 3.7. A partial map $f : A \rightharpoonup A$ is *represented* by $a \in A$ when, for all $b \in B$, $f(b) = a{\cdot}b$.

Theorem 3.8. *Every general recursive map $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is represented in the following sense: there is $a \in A$ such that, for all $n \in \mathbb{N}$, if $f(n) \Downarrow$ then $\overline{f(n)} = a \cdot \overline{n}$.*

4 (Total) combinatory algebras

Definition 4.1. A *(total) combinatory algebra (CA)* is given by a carrier set A and a *total* binary operation $- \cdot - : A \times A \to A$, such that there are K, S satisfying the characteristic equations

$$\mathsf{K} \cdot x \cdot y = x \qquad \text{and} \qquad \mathsf{S} \cdot x \cdot y \cdot z = (x \cdot z) \cdot (y \cdot z).$$

Proposition 4.2. *Every combinatory algebra is a partial combinatory algebra.*

Proof. Simply reuse the total application as the partial one. □

5 Examples of PCAs

5.1 Free combinatory algebra

Definition 5.1. The *free combinatory algebra* is generated freely by the symbols K and S and formal binary application, quotiented by provable equality.

5.2 The graph model

Definition 5.2. A *listing* on a set A is a section $\mathsf{fromList} : \mathsf{List}\, A \to A$ and a retraction $\mathsf{toList} : A \to \mathsf{List}\, A$.

Suppose A is a set with a listing. It induces a pairing on A: $\langle x, y\rangle = \mathsf{fromList}\,[x, y]$ with corresponding projections. Each $x \in A$ may be seen as a finite subset of A,

$$\mathsf{toSet}\, x = \{y \mid y \in \mathsf{toList}\, x\}.$$

Definition 5.3. The *application* $- \cdot - : \mathcal{P}\, A \times \mathcal{P}\, A \to \mathcal{P}\, A$ is defined by

$$S \cdot T = \{x \in A \mid \exists y \in A.\, \mathsf{toSet}\, y \subseteq T \wedge \langle x, y\rangle \in S\}.$$

Theorem 5.4. *The set $\mathcal{P}\, A$ with the application as above is a combinatory algebra.*