Partial Combinatory Algebras

Codrin Iftode & Johannes Kool

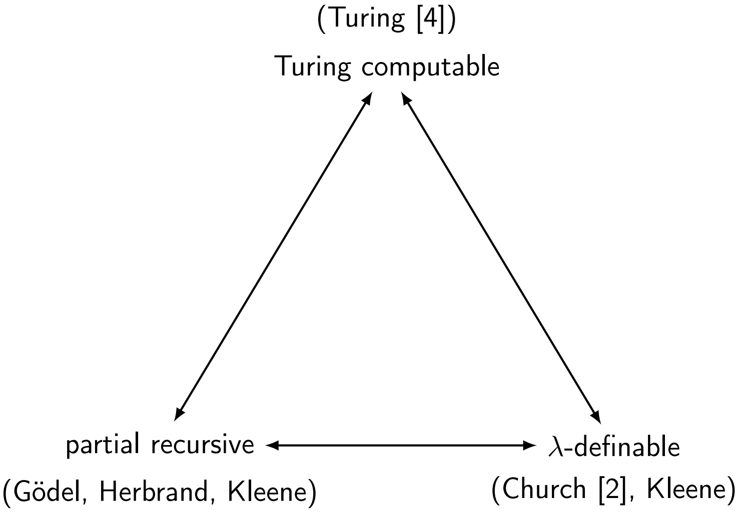
November 2025

Structure

We follow the 'Notes on realizability' by Andrej Bauer [1].

1. Motivation
2. Partial Combinatory Algebras (PCAs)
3. PCA as model of computation
4. Examples of PCAs

What is a computable function? - Church-Turing Thesis



Partial Recursive Functions, Informally

A **partial recursive function** [1] of the form $\mathbb{N}^k \to \mathbb{N}$ is built from the basic functions:

- Constant, e.g. $c(x, y, z) = 3$
- Projection, e.g. $p(x, y) = x$
- Successor, e.g. $S(x) = x + 1$

[1] Section 1.1 of [1]

Partial Recursive Functions, Informally

A **partial recursive function** [1] of the form $\mathbb{N}^k \to \mathbb{N}$ is built from the basic functions:

- Constant, e.g. $c(x, y, z) = 3$
- Projection, e.g. $p(x, y) = x$
- Successor, e.g. $S(x) = x + 1$

combined using the operators:

- Composition, e.g. $f \circ g$
- Primitive recursion, e.g.
  $f(0, x) = g(x)$
  $f(n + 1, x) = h(n, f(n, x), x)$
- Minimization, e.g. $\mu(f)(z, x)$ is the minimum z such that $f(z, x) = 0$.

[1] Section 1.1 of [1]

PCA's

- A set $\mathbb{A}$ with a partial binary operation $\cdot : \mathbb{A} \times \mathbb{A} \rightharpoonup \mathbb{A}$

PCA's

- A set $\mathbb{A}$ with a partial binary operation $\cdot : \mathbb{A} \times \mathbb{A} \rightharpoonup \mathbb{A}$
- $K \cdot x \cdot y = x$
- $S \cdot x \cdot y \cdot z \simeq (x \cdot z) \cdot (y \cdot z)$
  - ($S \cdot x \cdot y$ should be defined)

Notation

If a and b are possibly undefined expressions, we write $a \simeq b$ when either both a and b are undefined or both are defined and $a = b$.

Combinatory completeness

- S and K make the system combinatory complete.

Combinatory completeness

- S and K make the system combinatory complete.
  Intuition: generalise $(\lambda x.M)N =_\beta M[N/x]$

Combinatory completeness

- S and K make the system combinatory complete.
  Intuition: generalise $(\lambda x.M)N =_\beta M[N/x]$
- Expressions over $\mathbb{A}$: $e ::= x \mid a \in \mathbb{A} \mid e_1 \cdot e_2$

Combinatory completeness

- S and K make the system combinatory complete.
  Intuition: generalise $(\lambda x.M)N =_\beta M[N/x]$
- Expressions over $\mathbb{A}$: $e ::= x \mid a \in \mathbb{A} \mid e_1 \cdot e_2$
- For every variable x and expression e over $\mathbb{A}$, there is an expression e' over $\mathbb{A}$ whose variables are those of e excluding x such that $e' \downarrow$ and $e' \cdot a \simeq e[a/x]$ for all $a \in \mathbb{A}$.

Combinatory completeness

- S and K make the system combinatory complete.
  Intuition: generalise $(\lambda x.M)N =_\beta M[N/x]$
- Expressions over $\mathbb{A}$: $e ::= x \mid a \in \mathbb{A} \mid e_1 \cdot e_2$
- For every variable x and expression e over $\mathbb{A}$, there is an expression e' over $\mathbb{A}$ whose variables are those of e excluding x such that $e' \downarrow$ and $e' \cdot a \simeq e[a/x]$ for all $a \in \mathbb{A}$.

Proof.
We can construct such an expression e' and write it as $\langle x \rangle\ e$:

1. $\langle x \rangle\ x := SKK$
2. $\langle x \rangle\ y := Ky$ if y is a variable distinct from x
3. $\langle x \rangle\ a := Ka$ if $a \in \mathbb{A}$
4. $\langle x \rangle\ e_1\ e_2 := S(\langle x \rangle\ e_1)(\langle x \rangle\ e_2)$

□

- We can define all the functions we know and love from lambda calculus:

$$\begin{aligned} \text{pair} &:= \langle x\ y\ z\rangle\ z\ x\ y & \text{if} &:= \langle x\rangle\ x \\ \text{fst} &:= \langle p\rangle\ p\ (\langle x\ y\rangle\ y) & \text{true} &:= \langle x\ y\rangle\ x \\ \text{snd} &:= \langle p\rangle\ p\ (\langle x\ y\rangle\ y) & \text{false} &:= \langle x\ y\rangle\ y \end{aligned}$$

$$\text{if true}\ \ a\ b \simeq a$$

- We can define all the functions we know and love from lambda calculus:

$$
\begin{aligned}
\mathsf{pair} &:= \langle x\ y\ z\rangle\ z\ x\ y & \quad \mathsf{if} &:= \langle x\rangle\ x \\
\mathsf{fst} &:= \langle p\rangle\ p\ (\langle x\ y\rangle\ y) & \quad \mathsf{true} &:= \langle x\ y\rangle\ x \\
\mathsf{snd} &:= \langle p\rangle\ p\ (\langle x\ y\rangle\ y) & \quad \mathsf{false} &:= \langle x\ y\rangle\ y
\end{aligned}
$$

$$\mathsf{if}\ \mathsf{true}\ \ a\ b \simeq a$$

- Without the bracket notation from the previous slide it would look like this:

$$
\begin{aligned}
\mathtt{pair} = {} & \mathrm{S(S(KS)(S(S(KS)(S(KK)(KS)))(S(S(KS)(S(S(KS)(S(KK)(KS)))} \\
& \mathrm{(S(S(KS)(S(S(KS)(S(KK)(KS)))(S(KK)(KK))))(S(KK)(KK)))))} \\
& \mathrm{(S(S(KS)(S(KK)(KK)))(S(KK)(SKK))))))(S(S(KS)(S(KK)(KK)))} \\
& \mathrm{(S(S(KS)(KK))(KK))).}
\end{aligned}
$$

Figure: Bauer, A (2025)

Natural Numbers - Curry Numerals

$$\overline{0} := I = SKK \quad \overline{n+1} := \text{pair false}\,\overline{n}$$

Natural Numbers - Curry Numerals

$$\overline{0} := I = SKK \quad \overline{n+1} := \text{pair false}\,\overline{n}$$

$$\begin{aligned} \overline{0} &= I \\ \overline{1} &= (\text{false}, I) \\ \overline{2} &= (\text{false}, (\text{false}, I)) \\ &\dots \end{aligned}$$

Natural Numbers - Curry Numerals

$$\overline{0} := I = SKK \quad \overline{n+1} := \text{pair false}\,\overline{n}$$

$$\begin{aligned}\overline{0} &= I\\ \overline{1} &= (\text{false}, I)\\ \overline{2} &= (\text{false}, (\text{false}, I))\\ &\dots\end{aligned}$$

$$\begin{aligned}\text{succ} &:= \langle x\rangle\, \text{pair false}\, x\\ \text{iszero} &:= \text{fst}\\ \text{pred} &:= \langle x\rangle\, \text{if }\, (\text{iszero}\, x)\ \overline{0}\, \text{else}(\text{snd}\ \ x)\end{aligned}$$

Primitive Recursion

We can encode the Turing combinator [5]:

$$Y = (\langle x\ y\rangle\ y\ (x\ x\ y))\ (\langle x\ y\rangle\ y\ (x\ x\ y))$$

with $Yf \simeq f\ (Yf)$.

If $F = \langle f\ x\rangle\ \mathtt{if}\ (\mathtt{iszero}\ x)\ 0\ (x + f(\mathtt{pred}\ x))$, then

$$\begin{aligned} YF\ 3 &\simeq F\ (YF)\ 3 \\ &\simeq \mathtt{if}\ (\mathtt{iszero}\ 3)\ 0\ (3 + YF\ 2) \\ &\simeq 3 + YF\ 2 \\ &\simeq 3 + 2 + YF\ 1 \\ &\simeq 3 + 2 + 1 + 0 \end{aligned}$$

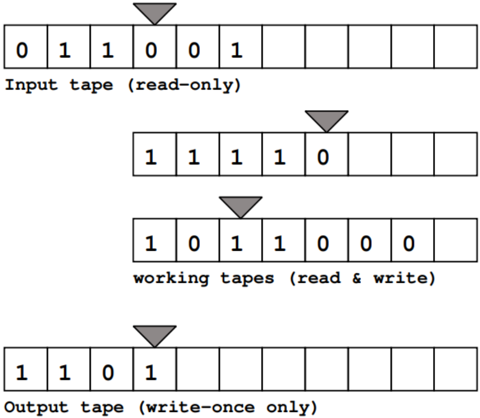
PCA models computation

In any PCA, we can encode natural numbers, pairs, conditionals, recursion, and minimization, and so we can encode any **partial recursive function**.

Conclusion

Any PCA is a model of computation!

Type 1 Turing Machines[2]



- Have **finite** input and output (tapes are infinite).
- Compute functions of the form $\mathbb{N} \rightharpoonup \mathbb{N}$.

[2]Sections 2.1.1 and 2.5.1 of [1]

Gödel Numbering

We can encode every Turing machine (TM) as a unique natural number.

For $x, y \in \mathbb{N}$, write $\varphi_x(y)$ for the output of executing the TM encoded by x on input encoded by y.

TM Theorems

Theorem (utm)

There exists a partial computable function $u : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ *such that* $u(x, y) \simeq \varphi_x(y)$ *for all* $x, y \in \mathbb{N}$.

Intuition: u is the universal Turing machine.

TM Theorems

Theorem (utm)

There exists a partial computable function $u : \mathbb{N} \times \mathbb{N} \rightharpoonup \mathbb{N}$ *such that* $u(x, y) \simeq \varphi_x(y)$ *for all* $x, y \in \mathbb{N}$*.*

Intuition: u is the universal Turing machine.

Theorem (simplified smn)

For every computable function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$*, there is a total computable function* $g : \mathbb{N} \to \mathbb{N}$ *such that* $f(x, y) \simeq \varphi_{g(x)}(y)$ *for all* $x, y, z \in \mathbb{N}$*.*

Intuition: we can "curry" Turing machines.

Example: Kleene's First Algebra - Type 1 TMs (1)

- $\mathbb{A} := \mathbb{N}$
- $n \cdot m := \varphi_n(m)$
- The function $p(x, y) = x$ is computable.

$$\begin{aligned} K \cdot x \cdot y &\simeq \varphi_{\varphi_K(x)}(y) \\ &\simeq \varphi_{q(x)}(y) \\ &\simeq p(x, y) \\ &= x \end{aligned}$$

Example: Kleene's First Algebra - Type 1 TMs (1)

- $\mathbb{A} := \mathbb{N}$
- $n \cdot m := \varphi_n(m)$
- The function $p(x, y) = x$ is computable.

$$
\begin{aligned}
K \cdot x \cdot y &\simeq \varphi_{\varphi_K(x)}(y) \\
&\simeq \varphi_{q(x)}(y) \\
&\simeq p(x, y) \\
&= x
\end{aligned}
$$

Define K as any natural number such that $\varphi_K = q$, where we get the computable $q : \mathbb{N} \to \mathbb{N}$ by "currying" p via the smn theorem.

Example: Kleene's First Algebra - Type 1 TMs (2)

- The function $g(x, y, z) = (x \cdot z) \cdot (y \cdot z)$ is computable (apply utm repeatedly).

$$
\begin{aligned}
S \cdot x \cdot y \cdot z &\simeq \varphi_{S \cdot x \cdot y}(z) \\
&\simeq \varphi_{\varphi_{S \cdot x}(y)}(z) \\
&\simeq \varphi_{\varphi_{\varphi_S(x)}(y)}(z) \\
&\simeq \varphi_{\varphi_{q(x)}(y)}(z) \\
&\simeq \varphi_{r(x,y)}(z) \\
&\simeq g(x, y, z) \\
&= (x \cdot z) \cdot (y \cdot z)
\end{aligned}
$$

Example: Kleene's First Algebra - Type 1 TMs (2)

- The function $g(x, y, z) = (x \cdot z) \cdot (y \cdot z)$ is computable (apply utm repeatedly).

$$
\begin{aligned}
S \cdot x \cdot y \cdot z &\simeq \varphi_{S \cdot x \cdot y}(z) \\
&\simeq \varphi_{\varphi_{S \cdot x}(y)}(z) \\
&\simeq \varphi_{\varphi_{\varphi_S(x)}(y)}(z) \\
&\simeq \varphi_{\varphi_{q(x)}(y)}(z) \\
&\simeq \varphi_{r(x,y)}(z) \\
&\simeq g(x, y, z) \\
&= (x \cdot z) \cdot (y \cdot z)
\end{aligned}
$$

Define S as any natural number such that $\varphi_S = q$, where we "curry" g to get a computable $r : \mathbb{N}^2 \to \mathbb{N}$, and "curry" r to get the computable $q : \mathbb{N} \to \mathbb{N}$.

Example: Untyped lambda calculus [3]

$$t ::= x \mid t_1 \ t_2 \mid \lambda x.t$$

- $\mathbb{A}$ is the set of **closed** lambda terms, quotiented by β-equivalence.
- $[x] \cdot [y] := [x \ y]$, i.e. the equivalence class of x applied to y.
- $K := [\lambda xy.x]$.
- $S := [\lambda xyz.(xz)(yz)]$.

[3] Sections 2.3 and 2.5.1 of [1]

Example: Kleene's Second Algebra - Type 2 TMs [4]

Define a function space $\mathbb{B} := \mathbb{N}^{\mathbb{N}}$. It forms a **Baire space**.

42	13	17	42	...
0	**1**	**2**	**3**	...

[4] Sections 2.1.2 and 2.5.1 of [1]

Example: Kleene's Second Algebra - Type 2 TMs [4]

Define a function space $\mathbb{B} := \mathbb{N}^{\mathbb{N}}$. It forms a **Baire space**.

42	13	17	42	...
0	**1**	**2**	**3**	...

	Type 1 TM	**Type 2 TM**
input/output	finite	infinite
computable function	$\mathbb{N} \rightharpoonup \mathbb{N}$	$\mathbb{B} \rightharpoonup \mathbb{B}$
encoding	$\mathbb{N}$	$\mathbb{B}$

[4] Sections 2.1.2 and 2.5.1 of [1]

Example: Kleene's Second Algebra - Type 2 TMs [4]

Define a function space $\mathbb{B} := \mathbb{N}^{\mathbb{N}}$. It forms a **Baire space**.

42	13	17	42	...
0	**1**	**2**	**3**	...

	Type 1 TM	**Type 2 TM**
input/output	finite	infinite
computable function	$\mathbb{N} \rightharpoonup \mathbb{N}$	$\mathbb{B} \rightharpoonup \mathbb{B}$
encoding	$\mathbb{N}$	$\mathbb{B}$

If $x, y \in \mathbb{B}$, write $\eta_x(y)$ for the output of running the type 2 machine encoded by x on input encoded by y.

Then, $(\mathbb{B}, \eta)$ forms a PCA, using the smn and utm theorems for type 2 TMs.

[4] Sections 2.1.2 and 2.5.1 of [1]

Example: Combinatory Logic

$$t ::= K \mid S \mid t \cdot t$$

Let $\approx$ be the least congruence relation ($a \cdot b \approx\ a' \cdot b'$ when $a \approx a'$ and $b \approx\ b'$) on the set CL, for all $a, b, c \in CL$,

$$K\ a\ b \approx a$$

$$S\ a\ b\ c \approx (a\ c)(b\ c)$$

The quotient $CL/\approx$ is the carrier of a total combinatory algebra $\mathbb{CL}$ called combinatory logic.

Other examples

- **Oracle Turing machines**: Turing machines with a infinite binary sequence $\omega : \mathbb{N} \to \{0, 1\}$ called an oracle, provided on a separate (infinite) input tape.

Other examples

- **Oracle Turing machines**: Turing machines with a infinite binary sequence $\omega : \mathbb{N} \to \{0, 1\}$ called an oracle, provided on a separate (infinite) input tape.
- **Infinite-time Turing machines** [3]: Turing machines that can run infinitely long.

Other examples

- **Oracle Turing machines**: Turing machines with a infinite binary sequence $\omega : \mathbb{N} \to \{0, 1\}$ called an oracle, provided on a separate (infinite) input tape.
- **Infinite-time Turing machines** [3]: Turing machines that can run infinitely long.
- **Reflexive domains**: topological models for the untyped lambda calculus.

Conclusion

Bibliography

[1] Bauer, A. (2025). Notes on realizability. Unpublished manuscript. https://www.andrej.com/zapiski/MGS-2022/notes-on-realizability.pdf. Accessed: Nov 12, 2025.

[2] Church, A. (1932). A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366.

[3] Hamkins, J. D. and Lewis, A. (2000). Infinite time turing machines. *Journal of Symbolic Logic*, 65(2):567–604.

[4] Turing, A. M. (1937a). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265. _eprint: https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s2-42.1.230.

[5] Turing, A. M. (1937b). The þ-function in λ-k-conversion. *Journal of Symbolic Logic*, 2(4):164–164.