How to define things by recursion

Alex Kavvos

LOGSEM seminar, 11 May 2020

Institut for Datalogi, Aarhus Universitet
Defining functions by recursion

Let

\[ \text{fact} : \mathbb{N} \rightarrow \mathbb{N} \]

\[ \text{fact}(n) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } n = 0 \\ n \times \text{fact}(n - 1) & \text{else} \end{cases} \]

Is this well-defined?

Operational solution: write an interpreter.

Mathematical solutions:

1. Postulate that “definition by induction” is a thing.
   But what about the following function?

   \[ f(n) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } n = 1 \\ \text{elsif } \text{even}(n) \text{ then } f(n/2) & \text{else} \ f(3n+1) \end{cases} \]

2. Write down a little abstract machine. (Implicitly just like 1.)

3. Do a little bit of domain theory. Fun for the whole family!
Embracing higher-order functions

Use \( \lambda \)-abstraction:

\[
\text{fact} = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times \text{fact}(n - 1)
\]

A very common form: a function defined in terms of itself. Abstract the recursive call:

\[
\text{fact} = (\lambda f. \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1)) \text{fact}
\]

This is of the form \( \text{fact} = F(\text{fact}) \), where

\[
F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})
\]

\[
F(f) = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1)
\]

where \( \mathbb{N} \rightarrow \mathbb{N} \) is the set of partial functions on \( \mathbb{N} \).

\[
\text{fact is a fixed point of } F.
\]
$$F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$$

$$F(f) = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n-1)$$

A curious phenomenon. If $$\bot : \mathbb{N} \rightarrow \mathbb{N}$$ is the undefined function, let

$$f_0 \overset{\text{def}}{=} \bot \overset{\text{def}}{=} \emptyset$$

$$f_{n+1} \overset{\text{def}}{=} F(f_n)$$

Observe that

$$f_1 = \{(0, 1)\}$$

$$f_2 = \{(0, 1), (1, 1)\}$$

$$f_3 = \{(0, 1), (1, 1), (2, 2)\}$$

$$f_4 = \{(0, 1), (1, 1), (2, 2), (3, 6)\}$$

$$\vdots$$
$F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$

$F(f) = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1)$

$f_0 = \emptyset$

$f_1 = \{(0, 1)\}$

$f_2 = \{(0, 1), (1, 1)\}$

$f_3 = \{(0, 1), (1, 1), (2, 2)\}$

$f_4 = \{(0, 1), (1, 1), (2, 2), (3, 6)\}$

Intuitively, fact is the **limit** of this sequence. Some observations:

1. $f_{i+1}$ is **consistent** with $f_i$.
2. $f_{i+1}$ is **more defined** than $f_i$. 
The Subset Order

1. $f_{i+1}$ is **consistent** with $f_i$.
2. $f_{i+1}$ is **more defined** than $f_i$.

Recall the *subset relation* between partial functions:

\[ f \subseteq g \; \text{def} \equiv \forall x, y \in \mathbb{N}. \; (x, y) \in f \implies (x, y) \in g \]

$g$ is possibly more defined than $f$, and agrees with it wherever both are defined. Writing $E \simeq E'$ for *Kleene equality*:

\[ f \subseteq g \; \text{def} \equiv \forall x, y \in \mathbb{N}. \; f(x) \simeq y \implies g(x) \simeq y \]

$\subseteq$ is a relation on $(\mathbb{N} \to \mathbb{N})$. It is a **partial order**:

- **reflexive** $f \subseteq f$
- **transitive** $f \subseteq g \land g \subseteq h \implies f \subseteq h$
- **antisymmetric** $f \subseteq g \land g \subseteq f \implies f = g$
Notice that $F(f) = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1)$ is **monotonic**: a more defined input leads to a more defined output.

\[
f \subseteq g \implies F(f) \subseteq F(g)
\]

We prove by induction that the sequence

\[
f_0 \overset{\text{def}}{=} \bot \quad \quad \quad f_{n+1} \overset{\text{def}}{=} F(f_n)
\]

is an $\omega$-chain:

\[
f_0 \subseteq f_1 \subseteq f_2 \subseteq f_3 \subseteq \ldots
\]

**BC:** $f_0 \overset{\text{def}}{=} \emptyset \subseteq f_1$ whatever $f_1$ is.

**IS:** if $f_i \subseteq f_{i+1}$ then $f_{i+1} = F(f_i) \subseteq F(f_{i+1}) = f_{i+2}$ by monotonicity.
Recap. To define the factorial function:

1. We characterised it as the **fixed point** of
   \[ F(f) = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1). \]
2. We produced a sequence \((f_i)_{i \in \omega}\) of **approximations** to it.
3. These approximations have a **sense of purpose**: they become progressively more defined, without contradicting previous information.

If we take the set

\[ f \overset{\text{def}}{=} \bigcup_{i \in \omega} f_i \]

we find that it is a **partial function** itself. (Why?)

\[ f \text{ is the limit of the sequence } (f_i)_{i \in \omega} \]
It remains to prove that $f \overset{\text{def}}{=} \bigcup_{i \in \omega} f_i$ is a fixed point of $F$.

$$F(f) = F \left( \bigcup_{i \in \omega} f_i \right) = ???$$

Something is missing.

- $f = \bigcup_{i \in \omega} f_i$ is a huge object: it is defined at all natural numbers.
- But $F(f)(n) \overset{\text{def}}{=} \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1)$ uses the value of $f$ at a finite number of points.

$F$ does not make any “decisions” based on the entirety of $f$.

We say that $F$ is continuous.
Continuity

**Definition**
A monotonic functional $F : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$ is **continuous** if for any $\omega$-chain $(f_i)_{i \in \omega}$ we know that

$$F \left( \bigcup_{i \in \omega} f_i \right) = \bigcup_{i \in \omega} F(f_i)$$

By monotonicity, we always have $\bigcup_{i \in \omega} F(f_i) \subseteq F \left( \bigcup_{i \in \omega} f_i \right)$. It suffices to check

$$F \left( \bigcup_{i \in \omega} f_i \right) \subseteq \bigcup_{i \in \omega} F(f_i)$$

That is: $F$ cannot make any decisions based on the whole limit!

**Example**
$F(f) = \text{if } (f = \text{id}_\mathbb{N}) \text{ then } \lambda n. 1 \text{ else } \lambda n. 0$ is not continuous.
The functional

\[ F(f) = \lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1) \]

is “obviously” continuous: it uses \( f \) at a finite number of points.

It remains to prove that \( f \overset{\text{def}}{=} \bigcup_{i \in \omega} f_i \) is a fixed point of \( F \).

\[
F(f) = F \left( \bigcup_{i \in \omega} f_i \right) = \bigcup_{i \in \omega} F(f_i) = \bigcup_{i \in \omega} f_{i+1} = \bigcup_{i \in \omega} f_i
\]

(The first term of an \( \omega \)-chain can be skipped in the union.)

So \( f \) is a fixed point.

We may take it as the definition of the factorial function.
Let \( g : \mathbb{N} \rightarrow \mathbb{N} \) be a total computable function.

Given the Gödel number \( \langle M \rangle \) of a Turing machine \( M \), read \( g(\langle M \rangle) \) as the Gödel number \( \langle N \rangle \) of another TM \( N \) (quite possibly gibberish).

Suppose \( g \) is extensional: if TMs \( M \) and \( N \) compute the same function, then so do the TMs encoded by \( g(\langle M \rangle) \) and \( g(\langle N \rangle) \).

**Example**
The function that writes out the source code of
\[
\lambda n. \text{if } n = 0 \text{ then } 1 \text{ else } n \times f(n - 1)
\]
when given the source of \( f \).

This defines a functional
\[
F_g : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})
\]

We call this an effective operation.
Theorem (Myhill & Sherpherdson, 1955)
Every effective operation $F_g : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N} \rightarrow \mathbb{N})$ is

1. monotonic
2. continuous
3. effective on finite functions

Moreover, every such functional is an effective operation.

The last condition means: there is a program that given the full list of input-output pairs of a finite function
\[
\theta \overset{\text{def}}{=} \{(x_1, y_1), \ldots, (x_n, y_n)\}
\] and some input $x$ computes $F(\theta)(x)$.

Thus, any reasonable template/specification has a fixed point. (Reasonable = there is a TM that when given code meant to run at the time of a recursive call outputs code for the entire function definition.)
Save the last bit, nothing so far depends on partial functions.

Let \( \sqsubseteq \) be a **partial order** on a set \( D \): a reflexive, transitive, antisymmetric relation. The following is akin to a **limit**.

**Definition (Least upper bound)**
The *least upper bound* of \( S \subseteq D \) is an element \( \bigcup S \in D \) such that

1. \( \forall x \in S. \ x \sqsubseteq \bigcup S \)
2. if \( \forall x \in S. \ x \sqsubseteq z \) then \( \bigcup S \sqsubseteq z \)

**Example**
Let \( \mathcal{W} \subseteq \mathcal{P}(X) \). The least upper bound of \( \mathcal{W} \) in \( (\mathcal{P}(X), \subseteq) \) is given by the **union**

\[
\bigcup \mathcal{W} \overset{\text{def}}{=} \{ x \in X \mid \exists S \in \mathcal{W}. \ x \in S \}
\]

It is the **least** set that contains all the sets in \( \mathcal{W} \).
\( \omega \)-complete partial orders

**Definition (\( \omega \)-complete partial order)**
A partial order \( (D, \sqsubseteq) \) is \( \omega \)-complete just if

1. it has a least element \( \bot \), so that \( \forall x \in D. \bot \sqsubseteq x \)
2. every \( \omega \)-chain \( (x_i)_{i \in \omega} \) has a least upper bound \( \bigsqcup_{i \in \omega} x_i \in D \).

Let \( D \) and \( E \) be \( \omega \)-cpos.

**Definition**
A function \( f : D \to E \) is **monotonic** if \( x \sqsubseteq y \implies f(x) \sqsubseteq f(y) \)

**Definition**
A function \( f : D \to E \) is **continuous** if for every \( \omega \)-chain \( (x_i)_{i \in \omega} \)

\[
f \left( \bigsqcup_{i \in \omega} x_i \right) = \bigsqcup_{i \in \omega} f(x_i)
\]
The Fixed Point Theorem

**Theorem (Kleene ≈1935, Tarski 1939)**

Let $f : D \rightarrow D$ be a continuous function on an $\omega$-cpo $D$. Then $f$ has a least fixed point given by

$$\text{lfp}(f) \overset{\text{def}}{=} \bigsqcup_{i \in \omega} f^i(\bot)$$

**Proof.**
Induction: $(f^i(\bot))_{i \in \omega}$ is an $\omega$-chain. The lub is a fixed point:

$$f \left( \bigsqcup_{i \in \omega} f^i(\bot) \right) = \bigsqcup_{i \in \omega} f(f^i(\bot)) = \bigsqcup_{i \in \omega} f^{i+1}(\bot) = \bigsqcup_{i \in \omega} f^i(\bot)$$

It is the least one. Suppose $f(x) = x$. Then $f^k(\bot) \sqsubseteq x$ by induction. So $x$ is an upper bound for $(f^k(\bot))_{i \in \omega}$. Hence $\bigsqcup_{i \in \omega} f^i(\bot) \sqsubseteq x$. \qed
Examples of $\omega$-cpos

powersets $(\mathcal{P}(X), \subseteq)$. Least upper bounds = unions.

partial functions $(\mathbb{N} \to \mathbb{N}, \subseteq)$. A sub-cpo of $\mathcal{P}(\mathbb{N} \times \mathbb{N})$.

flat nats $\mathbb{N}_\bot \overset{\text{def}}{=} \{\bot\} \cup \mathbb{N}$. $x \subseteq y \overset{\text{def}}{=} x = \bot \lor x = y$

$\omega$-chain are of two forms:

\[
\bot \subseteq \bot \subseteq \bot \subseteq \ldots \text{ (with lub } \bot) \\
\bot \subseteq \bot \subseteq \ldots \subseteq n \subseteq n \subseteq \ldots \text{ (with lub } n) \\
\]

streams $\Sigma^\infty \overset{\text{def}}{=} \text{finite or infinite sequences}$ over $\Sigma$. $w \subseteq v$ iff $w$ is a prefix of $v$. An $\omega$-chain over $\Sigma = \mathbb{B} \overset{\text{def}}{=} \{0, 1\}$:

\[
\epsilon \subseteq \langle 0 \rangle \subseteq \langle 0, 0 \rangle \subseteq \langle 0, 0, 0 \rangle \subseteq \ldots \\
\text{Lub: the infinite sequence } 0^\omega.
\]
Examples of monotonic and continuous functions

Flat booleans: $\mathbb{B}_\bot \overset{\text{def}}{=} \{ \bot \} \cup \mathbb{B}$. $x \sqsubseteq y \overset{\text{def}}{=} x = \bot \lor x = y$

Define three functions $f_1, f_2, f_3 : \mathbb{B}^\infty \to \mathbb{B}_\bot$.

$f_1(w) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } w \text{ contains a } 1 \\ \bot & \text{otherwise} \end{cases}$

$f_2(w) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } w \text{ contains a } 1 \\ 0 & \text{if } w = 0^\omega \\ \bot & \text{otherwise} \end{cases}$

$f_3(w) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } w \text{ contains a } 1 \\ 0 & \text{otherwise} \end{cases}$

• $f_1$ is continuous: it examines the stream element-by-element.
• $f_2$ is monotonic but not continuous: it makes a decision by looking at the entirety of an infinite stream.
• $f_3$ is just awful.
Goal: mathematically defining functions by recursion.

Main ideas:

- recursive definitions as **fixed points**
- the **partial order of definedness**
- **least upper bounds** (lub) as limits/completed objects
- **monotonic and continuous** functions as (i) computational functions and (ii) acceptable templates for recursive definitions
- the **fixed point theorem**: constructing fixed points by iterating continuous functions (can be generalised: if just monotone, we can iterate transfinitely).
Beyond

- The semantics of **PCF**: simply-typed $\lambda$-calculus + recursion.
- **Program logics** for recursion: computational induction.
- So far: convergence. But equally important is approximation. For example, partial functions are **algebraic**: $f = \bigcup_{\theta \subseteq \text{fin}} f \theta$. $\omega$-cpos that are continuous, algebraic, ... 
- Semantics of **recursive types**. In Haskell:
  
  ```haskell
  data Tree = Leaf Int | Node Tree Int Tree
  ```

  Must construct a mathematical ‘space’ $X$ that provides a solution to the **recursive domain equation**

  $$X \cong \mathbb{N}_\bot \oplus (X \times \mathbb{N}_\bot \times X)_\bot$$

- **Information Systems**: an equivalent presentation.
- **Synthetic Domain Theory**: a closer connection with computability.
There is another way: **take step-indexing seriously**. Replace \( \omega \)-cpos with sets **constructed over time**:

\[
P = P(0) \xleftarrow{r_0} P(1) \xleftarrow{r_1} P(2) \xleftarrow{r_2} \ldots
\]

\( P(i) \) = values at time \( i \). \( r_i : P(i+1) \to P(i) \) **trims** values.

Delaying a computation:

\[
\Rightarrow P = \{\ast\} \xleftarrow{1} P(0) \xleftarrow{r_0} P(1) \xleftarrow{r_1} P(2) \xleftarrow{r_2} \ldots
\]

A **causal** function \( f : P \to Q \) consists of a family \( f_i : P(i) \to Q(i) \) of functions that is ‘compatible’ with trimming.

**Theorem**

*Every causal function \( f : \Rightarrow P \to P \) has a guarded fixed point.*

Often just as good as domain theory. Excellent for recursive types!
This presentation is based on lecture notes by Samson Abramsky. (≈2007).

The history of the fixed point theorem:


Standard references on domain theory—a book and a survey:


Possibly the most clear and concise reference to PCF/LCF:

A really unusual and fascinating book on (a) the connections of domain theory with topology, and (b) the intuitive meanings of many domain-theoretic and topological concepts in Haskell:


A very similar blog post:

The source of all synthetic guarded domain theory: