

# Advanced Topics in Theoretical Computer Science

## Part 3: Recursive Functions (3)

14.12.2022

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: [sofronie@uni-koblenz.de](mailto:sofronie@uni-koblenz.de)

# Contents

---

- Recapitulation: Turing machines and Turing computability
- Register machines (LOOP, WHILE, GOTO)
- **Recursive functions**
- The Church-Turing Thesis
- Computability and (Un-)decidability
- Complexity
- Other computation models: e.g. Büchi Automata,  $\lambda$ -calculus

# 3. Recursive functions

---

- Introduction/Motivation
- Primitive recursive functions  $\mapsto \mathcal{P}$
- $\mathcal{P} = \text{LOOP}$
- $\mu$ -recursive functions  $\mapsto F_\mu$
- $F_\mu = \text{WHILE}$
- Summary

# Primitive recursive functions

---

## Definition (Primitive recursive functions)

- **Atomic functions:** The functions
  - Null 0
  - Successor +1
  - Projection  $\pi_i^k$  ( $1 \leq i \leq k$ )are primitive recursive.
- **Composition:** The functions obtained by composition from primitive recursive functions are primitive recursive.
- **Primitive recursion:** The functions obtained by primitive recursion from primitive recursive functions are primitive recursive.

The set of all primitive recursive functions is the smallest set with the properties above.

**Notation:**  $\mathcal{P} =$  The set of all primitive recursive functions

# Bounded $\mu$ operator

**Definition.** Let  $g : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  be a function.

The **bounded  $\mu$  operator** is defined as follows:

$$\mu_{i < m} i (g(\mathbf{n}, i) = 0) := \begin{cases} i_0 & \text{if } g(\mathbf{n}, i_0) = 0, \quad i_0 < m \\ & \text{and for all } j < i_0 \quad g(\mathbf{n}, j) \neq 0 \\ 0 & \text{if } g(\mathbf{n}, j) \neq 0 \text{ for all } 0 \leq j < m \\ & \text{or } m = 0 \end{cases}$$

$\mu_{i < m} i (g(\mathbf{n}, i) = 0)$  is the smallest  $i < m$  such that  $g(\mathbf{n}, i) = 0$

**Theorem.** If  $g : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  is a primitive recursive function then the function  $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  defined below is also primitive recursive.

$$f(\mathbf{n}, m) = \mu_{i < m} i (g(\mathbf{n}, i) = 0)$$

# Prime number functions

---

**Theorem:** The following functions are primitive recursive:

(1) The Boolean function  $| : \mathbb{N} \times \mathbb{N} \rightarrow \{0, 1\}$  defined by:

$$|(n, m) = \begin{cases} 1 & \text{if } n \text{ divides } m \\ 0 & \text{otherwise} \end{cases}$$

(2) The Boolean function  $\text{prime} : \mathbb{N} \rightarrow \{0, 1\}$  defined by:

$$\text{prime}(n) = \begin{cases} 1 & \text{if } n \text{ prime} \\ 0 & \text{otherwise} \end{cases}$$

(3) The function  $p : \mathbb{N} \rightarrow \mathbb{N}$  defined by:  $p(n) = p_n$ , the  $n$ -th prime number.

(4) The function  $D : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  defined by:  $D(n, i) = k$  iff  $k$  is the power of the  $i$ -th prime number in the prime number decomposition of  $n$ .

$$D(n, i) = \max(\{j \mid n \bmod p(i)^j = 0\})$$

# 3. Recursive functions

---

- Introduction/Motivation
- Primitive recursive functions  $\mapsto \mathcal{P}$
- $\mathcal{P} = \text{LOOP}$
- $\mu$ -recursive functions  $\mapsto F_\mu$
- $F_\mu = \text{WHILE}$
- Summary

# Goal

---

Show that  $\mathcal{P} = \text{LOOP}$

Idea:

To show that  $\mathcal{P} \supseteq \text{LOOP}$  we have to show that every LOOP computable function can be expressed as a primitive recursive function.

For this, we will encode the contents of arbitrarily many registers in one natural number (used as input for this primitive recursive function).

For this encoding we will use Gödelisation. We will need to show that Gödelisation is primitive recursive.

To show that  $\mathcal{P} \subseteq \text{LOOP}$  we have to show that:

- all atomic primitive recursive functions are LOOP computable, and
- LOOP is closed under composition of functions and primitive recursion.



# Gödelisation

---

To show: Gödelisierung is primitive recursive

Informally:

- Coding number sequences as a number
- Corresponding decoding function (projection)

are primitiv recursive

More precise formulation:

There exist primitive recursive functions

$$K^r : \mathbb{N}^r \rightarrow \mathbb{N} \quad (r \geq 1)$$

$$D_i : \mathbb{N} \rightarrow \mathbb{N} \quad (1 \leq i \leq r)$$

with:

$$D_i(K^r(n_1, \dots, n_r)) = n_i$$

# Gödelisation

---

To show: Gödelisation is primitive recursive

Informally:

- Coding number sequences as a number
- Corresponding decoding function (projection)

are primitive recursive

Recall:

Gödelisation: Coding number sequences as a number

Bijection  $K$  between  $\bigcup_{r \in \mathbb{N}} \mathbb{N}^r$  and  $\mathbb{N}$ :

$K^r : \mathbb{N}^r \rightarrow \mathbb{N}$ , defined by:

$$K^r(n_1, \dots, n_r) = \prod_{i=1}^r p(i)^{n_i}.$$

$n \in \mathbb{N}^r \mapsto K(n) = K_r(n)$

Decoding: The inverses  $D_i : \mathbb{N} \rightarrow \mathbb{N}$  of  $K^r$  defined by  $D_i(n) = D(n, i)$

# Gödelisation

---

Bijection  $K$  between  $\bigcup_{r \in \mathbb{N}} \mathbb{N}^r$  and  $\mathbb{N}$ :

$K^r : \mathbb{N}^r \rightarrow \mathbb{N}$ , defined by:

$$K^r(n_1, \dots, n_r) = \prod_{i=1}^r p(i)^{n_i}.$$

$$n \in \mathbb{N}^r \mapsto K(n) = K_r(n)$$

$D_i : \mathbb{N} \rightarrow \mathbb{N}$ ,  $1 \leq i \leq r$ , defined by  $D_i(n) = D(n, i)$

**Theorem.**  $K^r$  and  $D_1, \dots, D_r$  are primitive recursive.

**Lemma.**

(1)  $D_i(K^r(n_1, \dots, n_r)) = n_i$  for all  $1 \leq i \leq r$ .

(2)  $K^r(n_1, \dots, n_r) = K^{r+1}(n_1, \dots, n_r, 0)$

In general,  $D_i(K^r(n_1, \dots, n_r)) = 0$  if  $i > r$ .

# Gödelisation

---

## Notation:

$$K^r(n_1, \dots, n_r) = \langle n_1, \dots, n_r \rangle$$

$$D_i(n) = (n)_i$$

For  $r = 0$ :

$$\langle \rangle = 1$$

$$(\langle \rangle)_i = 0$$

# Gödelisation: Applications

---

## Theorem (Simultaneous Recursion)

If

$$f_1(\mathbf{n}, 0) = g_1(\mathbf{n})$$

...

$$f_r(\mathbf{n}, 0) = g_r(\mathbf{n})$$

$$f_1(\mathbf{n}, m + 1) = h_1(\mathbf{n}, m, f_1(\mathbf{n}, m), \dots, f_r(\mathbf{n}, m))$$

...

$$f_r(\mathbf{n}, m + 1) = h_r(\mathbf{n}, m, f_1(\mathbf{n}, m), \dots, f_r(\mathbf{n}, m))$$

and if  $g_1, \dots, g_r, h_1, \dots, h_r$  are primitive recursive  
then  $f_1, \dots, f_r$  are primitive recursive.

# Example

---

Let  $f_1$  and  $f_2$  be defined by simultaneous recursion as follows:

$$f_1(0) = 0$$

$$f_2(0) = 1$$

$$f_1(n + 1) = f_2(n)$$

$$f_2(n + 1) = f_1(n) + f_2(n)$$

# Example

---

Let  $f_1$  and  $f_2$  be defined by simultaneous recursion as follows:

$$f_1(0) = 0$$

$$g_1 = 0$$

$$f_2(0) = 1$$

$$g_2 = 1$$

$$f_1(n+1) = f_2(n)$$

$$h_1(n, f_1(n), f_2(n)) = f_2(n)$$

$$h_1 = \pi_3^3$$

$$f_2(n+1) = f_1(n) + f_2(n)$$

$$h_2(n, f_1(n), f_2(n)) = f_1(n) + f_2(n)$$

$$h_2 = + \circ (\pi_2^3, \pi_3^3)$$

# Gödelisation: Applications

---

## Theorem (Simultaneous Recursion)

If

$$f_1(\mathbf{n}, 0) = g_1(\mathbf{n})$$

...

$$f_r(\mathbf{n}, 0) = g_r(\mathbf{n})$$

$$f_1(\mathbf{n}, m + 1) = h_1(\mathbf{n}, m, f_1(\mathbf{n}, m), \dots, f_r(\mathbf{n}, m))$$

...

$$f_r(\mathbf{n}, m + 1) = h_r(\mathbf{n}, m, f_1(\mathbf{n}, m), \dots, f_r(\mathbf{n}, m))$$

and if  $g_1, \dots, g_r, h_1, \dots, h_r$  are primitive recursive  
then  $f_1, \dots, f_r$  are primitive recursive.



# Gödelisation: Applications

---

Proof: We define a new function  $f$  by:

$$f(\mathbf{n}, m) = \langle f_1(\mathbf{n}, m), \dots, f_r(\mathbf{n}, m) \rangle$$

$f$  can be computed by primitive recursion as follows:

$$\begin{aligned} f(\mathbf{n}, 0) &= \langle g_1(\mathbf{n}), \dots, g_r(\mathbf{n}) \rangle \\ f(\mathbf{n}, m + 1) &= \langle h_1(\mathbf{n}, m, (f(\mathbf{n}, m))_1, \dots, (f(\mathbf{n}, m))_r), \dots, \\ &\quad h_r(\mathbf{n}, m, (f(\mathbf{n}, m))_1, \dots, (f(\mathbf{n}, m))_r) \rangle \end{aligned}$$

$K^r \circ (g_1, \dots, g_r)$  and  $K^r \circ (h_1, \dots, h_r)$  are primitive recursive.

For all  $1 \leq i \leq r$ ,  $f_i(\mathbf{n}, m) = D_i(f(\mathbf{n}, m))$ .

Since  $f_i = D_i \circ f$  is primitive recursive, it follows that  $f_i$  is primitive recursive for all  $1 \leq i \leq r$ .

# Goal

---

Show that  $\mathcal{P} = \text{LOOP}$

Idea:

To show that  $\mathcal{P} \supseteq \text{LOOP}$  we have to show that every LOOP computable function can be expressed as a primitive recursive function.

For this, we will encode the contents of arbitrarily many registers in one natural number (used as input for this primitive recursive function).

For this encoding we use Gödelisation. We showed that Gödelisation is primitive recursive.

To show that  $\mathcal{P} \subseteq \text{LOOP}$  we have to show that:

- all atomic primitive recursive functions are LOOP computable, and
- LOOP is closed under composition of functions and primitive recursion.

$$\mathcal{P} = \text{LOOP}$$

---

**Theorem ( $\mathcal{P} = \text{LOOP}$ ).** The set of all LOOP computable functions is equal to the set of all primitive recursive functions

Proof (Idea)

1.  $\mathcal{P} \subseteq \text{LOOP}$

# $\mathcal{P} = \text{LOOP}$

---

**Theorem ( $\mathcal{P} = \text{LOOP}$ ).** The set of all LOOP computable functions is equal to the set of all primitive recursive functions

Proof (Idea)

## 1. $\mathcal{P} \subseteq \text{LOOP}$

- 1a: We show that all atomic primitive recursive functions are LOOP computable
- 1b: We show that LOOP is closed under composition of functions
- 1c: We show that LOOP is closed under primitive recursion

# $\mathcal{P} = \text{LOOP}$

---

**Theorem ( $\mathcal{P} = \text{LOOP}$ ).** The set of all LOOP computable functions is equal to the set of all primitive recursive functions

Proof (Idea)

## 1. $\mathcal{P} \subseteq \text{LOOP}$

1a: All atomic primitive recursive functions are LOOP computable

0 :	$x_1 := x_1 - 1$	//NOP
+1 :	$x_2 := x_1 + 1$	
$\pi_j^k$	$x_{k+1} := x_j$	

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **1b: LOOP is closed under composition of functions**

Let  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  with  $f(\mathbf{n}) = h(g_1(\mathbf{n}), \dots, g_r(\mathbf{n}))$

Assume that:

- $P_h$  computes  $h$
- $P_{g_j}$  computes  $g_j$  ( $1 \leq j \leq r$ )

**Idea:**  $f$  is computed by the program  $P_f$ :

$$P'_{g_1}; \dots; P'_{g_r}; P'_h$$

where  $P'_{g_i}$  differs from  $P_{g_i}$  (and  $P'_h$  from  $P_h$ ) only up to the fact that registers have been renamed/the contents stored in them copied.

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **1b: LOOP is closed under composition of functions**

Let  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  with  $f(\mathbf{n}) = h(g_1(\mathbf{n}), \dots, g_r(\mathbf{n}))$

Assume that:

- $P_h$  computes  $h$
- $P_{g_j}$  computes  $g_j$  ( $1 \leq j \leq r$ )

**More precisely:**  $P'_{g_i}$ : obtained from  $P_{g_i}$  by renaming register  $x_{k+i}$  to  $x_{k+r+i}$ .  
 $\mapsto$  keep free registers  $x_{k+1}, \dots, x_{k+r}$  for writing result of  $P_{g_1}, \dots, P_{g_r}$

$P'_h$ : obtained from  $P_h$  by renaming  $x_j$  to  $x_{k+j}$ .

$P_f :$   $P'_{g_1}; x_{k+1} := x_{k+r+1}; x_{k+r+1} := 0; \dots$

$P'_{g_r}; x_{k+r} := x_{k+r+1}; x_{k+r+1} := 0;$

$P'_h; x_{k+1} := x_{k+r+1}; x_{k+2} := 0; \dots; x_{k+r+1} := 0$

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **1c: LOOP is closed under primitive recursion**

Assume that  $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  is such that:

$$f(\mathbf{n}, 0) = g(\mathbf{n})$$

$$f(\mathbf{n}, m + 1) = h(\mathbf{n}, m, f(\mathbf{n}, m))$$

Then  $f$  is computed by the following LOOP Program:

```
xstorem := xk+1; // Number of loops (m)
xk+1 := 0; // Actual value of m (at the beginning 0)
P'g; // Computes f(n, 0); result in xk+2
loop xstorem do
  Ph; // Computes f(n, xk+1 + 1) = h(n, m, f(n, m))
  xk+2 := xk+2+1; // xk+2 = f(n, xk+1 + 1)
  xk+2+1 := 0;
  xk+1 := xk+1 + 1 // m = m + 1
end;
xstorem := 0
```



# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **1c: LOOP is closed under primitive recursion**

Assume that  $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  is such that:

$$f(\mathbf{n}, 0) = g(\mathbf{n})$$

$$f(\mathbf{n}, m + 1) = h(\mathbf{n}, m, f(\mathbf{n}, m))$$

Then  $f$  is computed by the following LOOP Program:

```
xstorem := xk+1; // Number of loops (m)
xk+1 := 0; // Actual value of m (at the beginning 0)
P'g; // Computes f(n, 0); result in xk+2
loop xstorem do
  Ph;
  xk+2 := xk+2+1;
  xk+2+1 := 0;
  xk+1 := xk+1 + 1
end;
xstorem := 0
```

where  $P'_g$  differs from  $P_g$  only in the fact that some registers have been renamed (e.g. output in  $x_{k+2}$ , not in  $x_{k+1}$ )

# $\mathcal{P} = \text{LOOP}$

---

**Theorem ( $\mathcal{P} = \text{LOOP}$ ).** The set of all LOOP computable functions is equal to the set of all primitive recursive functions

Proof (Idea)

## 2. $\text{LOOP} \subseteq \mathcal{P}$

Let  $P$  be a LOOP program which:

- uses registers  $x_1, \dots, x_l$
- has  $m$  loop instructions

We construct a primitive recursive function  $f_P$  which “simulates”  $P$

$$f_P(\langle n_1, \dots, n_l, h_1, \dots, h_m \rangle) = \langle n'_1, \dots, n'_l, h_1, \dots, h_m \rangle$$

if and only if:

$P$  started with  $n_i$  in register  $x_i$  terminates with  $n'_i$  in  $x_i$  ( $1 \leq i \leq l$ ).

In  $h_j$  it is “recorded” how long loop  $j$  should still run.

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd)

At the beginning and at the end of the simulation of  $P$  we have

$$h_1 = 0, \dots, h_m = 0.$$

Assume that we can construct a primitive recursive function  $f_P$  which “simulates”  $P$ , i.e.  $f_P(\langle n_1, \dots, n_l, h_1, \dots, h_m \rangle) = \langle n'_1, \dots, n'_l, h_1, \dots, h_m \rangle$

if and only if:

$P$  started with  $n_i$  in register  $x_i$  terminates with  $n'_i$  in  $x_i$  ( $1 \leq i \leq l$ ).

The function computed by the LOOP program  $P$  is then primitive recursive, since:

$$g(n_1, \dots, n_l) = g(n_1, \dots, n_k, 0, \dots, 0) = (f_P(\langle n_1, \dots, n_l, 0, 0, \dots \rangle))_{k+1}$$

(the input in registers  $x_1, \dots, x_k$ , all other registers contain 0, output in register  $x_{k+1}$ )

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **Construction of  $f_P$ :**

2a:  $P$  is  $x_i := x_i + 1$

$$f_P(n) = \langle (n)_1, \dots, (n)_{i-1}, (n)_i + 1, (n)_{i+1}, \dots \rangle = n * p(i)$$

$P$  is  $x_i := x_i - 1$

$$f_P(n) = \langle (n)_1, \dots, (n)_{i-1}, (n)_i - 1, (n)_{i+1}, \dots \rangle$$

$$f_P(n) = \begin{cases} n & D(n, i) = 0 \\ n \text{ DIV } p(i) & \text{otherwise} \end{cases}$$

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **Construction of  $f_P$ :**

2a:  $P$  is  $x_i := x_i + 1$

$$f_P(n) = \langle (n)_1, \dots, (n)_{i-1}, (n)_i + 1, (n)_{i+1}, \dots \rangle$$

$P$  is  $x_i := x_i - 1$

$$f_P(n) = \langle (n)_1, \dots, (n)_{i-1}, (n)_i - 1, (n)_{i+1}, \dots \rangle$$

2b:  $P$  is  $P_1; P_2$

$$f_P = f_{P_2} \circ f_{P_1} \quad \text{i.e. } f_P(n) = f_{P_2}(f_{P_1}(n))$$

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **Construction of  $f_P$ :**

2c:  $P$  is loop  $x_i$  do  $P_1$  end

Let  $f_{P_1}$  be the p.r. function which computes what  $P_1$  computes.

Initialize the  $j$ -th loop:

$$f_1(n) = \langle (n)_1, \dots, (n)_l, (n)_{l+1}, \dots, (n)_{l+j-1}, (n)_i, (n)_{l+j+1}, \dots \rangle$$

Let the  $j$ -th loop run:

$$f_2(n) = \begin{cases} n & \text{if } (n)_{l+j} = 0 \\ f_{P_1}(f_2(\langle \dots, (n)_{l+j} - 1, \dots \rangle)) & \text{otherwise} \end{cases}$$

Then:

$$f_P(n) = f_2(f_1(n)) = (f_2 \circ f_1)(n)$$

# $\mathcal{P} = \text{LOOP}$

---

Proof (ctd) **Construction of  $f_P$ :**

2c:  $P$  is loop  $x_i$  do  $P_1$  end

Let  $f_{P_1}$  be the p.r. function which computes what  $P_1$  computes.

Initialize the  $j$ -th loop:

$$f_1(n) = \langle (n)_1, \dots, (n)_l, (n)_{l+1}, \dots, (n)_{l+j-1}, (n)_i, (n)_{l+j+1}, \dots \rangle$$

$$f_1(n) = n * p(l+j)^{(n)_i}. \quad \text{if } (n)_{l+j} = 0 \text{ before the loop is executed}$$

Let the  $j$ -th loop run:

$$f_2(n) = \begin{cases} n & \text{if } (n)_{l+j} = 0 \\ f_{P_1}(f_2(n \text{ DIV } p(l+j))) & \text{otherwise} \end{cases}$$

Then:

$$f_P = f_2 \circ f_1$$

# $\mathcal{P} = \text{LOOP}$

---

**Proof** (ctd) We show that  $f_2$  is primitive recursive.

Let  $F : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  be defined by:

$$F(n, 0) = n$$

$$F(n, m + 1) = f_{P_1}(F(n, m))$$

Then  $F \in \mathcal{P}$ .

It can be checked that  $f_2(n) = F(n, D(n, l + j))$ . Therefore,  $f_2 \in \mathcal{P}$ .

Since  $f_1, f_2$  are primitive recursive, so is  $f_P = f_2 \circ f_1$ .



# 3. Recursive functions

---

- Introduction/Motivation
- Primitive recursive functions  $\mapsto \mathcal{P}$
- $\mathcal{P} = \text{LOOP}$
- $\mu$ -recursive functions  $\mapsto F_\mu$
- $F_\mu = \text{WHILE}$
- Summary