5. The Primitive Recursive Functions

- URM and TM based on universal programming languages.
- In this and the next section we introduce a third model of computation.
- It is given as a set of partial functions
  - basic functions
  - by using certain operations.
- First proposed by Gödel and Kleene 1936.
- Best model for showing that functions are computable.
- In this section we introduce the primitive-recursive functions, which form a subset of the partial-recursive functions.
Overview

(a) Introduction of **primitive recursive functions**.
   - Will be total.
   - Includes all functions which can be computed realistically, and many more.
   - But not all computable functions are primitive recursive.

(b) **Closure Properties of the primitive rec. functions**
   - We will show that the set of primitive recursive functions is a reach set of functions, closed under many operations.
Inductive definition of the **primitive recursive** functions 
\[ f : \mathbb{N}^k \to \mathbb{N}. \]

The following **basic Functions** are primitive recursive:

- zero : \( \mathbb{N} \to \mathbb{N} \),
- succ : \( \mathbb{N} \to \mathbb{N} \),
- proj\(_i^k\) : \( \mathbb{N}^k \to \mathbb{N} (0 \leq i < k) \).

Remember that these functions have defining equations

- \( \text{zero}(n) = 0 \),
- \( \text{succ}(n) = n + 1 \),
- \( \text{proj}_i^k(a_0, \ldots, a_{k-1}) = a_i \).
Def. Prim. Rec. Functions

If

- \( f : \mathbb{N}^k \to \mathbb{N} \) is primitive recursive,
- \( g_i : \mathbb{N}^n \to \mathbb{N} \) are primitive recursive, \( (i = 0, \ldots, k - 1) \),

so is

\[
f \circ (g_0, \ldots, g_{k-1}) : \mathbb{N}^n \to \mathbb{N}.
\]

Remember that \( h := f \circ (g_0, \ldots, g_{k-1}) \) is defined as

\[
h(\vec{x}) = f(g_0(\vec{x}), \ldots, g_{k-1}(\vec{x})).
\]

Especially, if \( f : \mathbb{N} \to \mathbb{N} \) and \( g : \mathbb{N} \to \mathbb{N} \) are primitive recursive, so is

\[
f \circ g : \mathbb{N} \to \mathbb{N}.
\]
Def. Prim. Rec. Functions

If

\[ g : \mathbb{N}^n \rightarrow \mathbb{N}, \]
\[ h : \mathbb{N}^{n+2} \rightarrow \mathbb{N} \]
are primitive recursive,

so is the function \( f : \mathbb{N}^{n+1} \rightarrow \mathbb{N} \) defined by primitive recursion from \( g, h \).

Remember that \( f \) is defined by

\[ f(\vec{x}, 0) = g(\vec{x}), \]
\[ f(\vec{x}, n + 1) = h(\vec{x}, n, f(\vec{x}, n)). \]

\( f \) is denoted by \( \text{primrec}(g, h) \).
Def. Prim. Rec. Functions

If

\[ k \in \mathbb{N}, \]
\[ h : \mathbb{N}^2 \rightarrow \mathbb{N} \] is primitive recursive,

so is the function \( f : \mathbb{N} \rightarrow \mathbb{N} \), defined by primitive recursion from \( k \) and \( h \).

Remember that \( f := \text{primrec}(k, h) \) is defined by

\[ f(0) = k, \]
\[ f(n + 1) = h(n, f(n)). \]

\( f \) is denoted by \( \text{primrec}(k, h) \).
Inductively Defined Sets

That the set of primitive recursive functions is inductively defined means:

- It is the least set containing basic functions and closed under the operations.
- Or: It is the set generated by the above.
- Or: The primitive recursive functions are those we can write as terms formed
  from zero, succ, proj\_i^n, using composition \( f \circ (g_0, \ldots, g_{n-1}) \)
  i.e. by forming from \( f, g_i \circ (g_0, \ldots, g_{n-1}) \)
  and primrec.
Inductively Defined Sets

E.g.

\[
\text{primrec} \left( \begin{array}{c}
\text{proj}_0^1, \\
\text{succ} \circ \text{proj}_2^3
\end{array} \right) : \mathbb{N}^2 \to \mathbb{N} \text{ is prim. rec.}
\]

\[
\begin{array}{c}
: \mathbb{N} \to \mathbb{N} \\
: \mathbb{N} \to \mathbb{N} \\
: \mathbb{N}^3 \to \mathbb{N}
\end{array}
\]

\[
: \mathbb{N}^2 \to \mathbb{N}
\]

(= addition)

\[
\text{primrec} \left( \begin{array}{c}
0, \\
\text{proj}_0^2
\end{array} \right) : \mathbb{N} \to \mathbb{N} \text{ is prim. rec.}
\]

\[
\begin{array}{c}
\in \mathbb{N} \\
: \mathbb{N}^2 \to \mathbb{N}
\end{array}
\]

\[
: \mathbb{N}^1 \to \mathbb{N}
\]

(= pred)

A relation $R \subseteq \mathbb{N}^n$ is **primitive recursive**, if

$$\chi_R : \mathbb{N}^n \to \mathbb{N}$$

is primitive recursive.

Note that we identified a set $A \subseteq \mathbb{N}^n$ with the relation $R \subseteq \mathbb{N}^n$ given by

$$R(\bar{x}) :\Leftrightarrow \bar{x} \in A$$

Therefore a set $A \subseteq \mathbb{N}^n$ is primitive recursive if the corresponding relation $R$ is.
Remark

Unless demanded explicitly, for showing that $f$ is defined by the principle of primitive recursion (i.e. by `primrec`), it suffices to express:

- $f(\vec{x}, 0)$ as an expression built from
  - previously defined prim. rec. functions,
  - $\vec{x}$,
  - and constants.

**Example:**

$$f(x_0, x_1, 0) = (x_0 + x_1) \cdot 3.$$  

(Assuming that $+, \cdot$ have already been shown to be primitive recursive).
Remark

- $f(\vec{x}, y + 1)$ as an expression built from 
  - previously defined prim. rec. functions, 
  - $\vec{x}$, 
  - the \textit{recursion argument} $y$, 
  - the \textit{recursion hypothesis} $f(\vec{x}, y)$, 
  - and constants.

**Example:**

$$f(x_0, x_1, y + 1) = (x_0 + x_1 + y + f(x_0, x_1, y)) \cdot 3 \ .$$

(Assuming that $+, \cdot$ have already been shown to be primitive recursive).
Remark

Similarly, for showing $f$ is prim. rec. by using previously defined functions using composition, it suffices to express $f(\vec{x})$ in terms of

- previously defined prim. rec. functions,
- parameters $\vec{x}$
- constants.

**Example:**

$$f(x, y, z) = (x + y) \cdot 3 + z.$$  

(Assuming that $+$, $\cdot$ have already been shown to be primitive recursive).

When looking at the first examples, we will express primitive recursive functions directly by using the basic functions, `primrec` and $\circ$. 
Identity Function

\( \text{id} : \mathbb{N} \to \mathbb{N}, \text{id}(n) = n \) is primitive recursive:

\[ \text{id} = \text{proj}_0^1: \]
\[ \text{proj}_0^1 : \mathbb{N}^1 \to \mathbb{N}, \]
\[ \text{proj}_0^1(n) = n = \text{id}(n). \]
Constant Function

\[ \text{const}_n : \mathbb{N} \to \mathbb{N}, \text{const}_n(k) = n \text{ is primitive recursive:} \]

\[ \text{const}_n = \text{succ} \circ \cdots \circ \text{succ} \circ \text{zero} : \]

\[ n \text{ times} \]

\[ \text{succ} \circ \cdots \circ \text{succ} \circ \text{zero}(k) = \text{succ}(\text{succ}(\cdots \text{succ}(\text{zero}(k)))) \]

\[ n \text{ times} \]

\[ = \text{succ}(\text{succ}(\cdots \text{succ}(0))) \]

\[ n \text{ times} \]

\[ = 0 + 1 + 1 \cdots + 1 \]

\[ n \text{ times} \]

\[ = n \]

\[ = \text{const}_n(k) . \]
Addition

\[ \text{add} : \mathbb{N}^2 \rightarrow \mathbb{N}, \text{add}(x, y) = x + y \]

is primitive recursive.

We have the laws:

\[
\begin{align*}
\text{add}(x, 0) &= x + 0 \\
&= x
\end{align*}
\]

\[
\begin{align*}
\text{add}(x, y + 1) &= x + (y + 1) \\
&= (x + y) + 1 \\
&= \text{add}(x, y) + 1
\end{align*}
\]
Addition

\[ \text{add}(x, 0) = x + 0, \]
\[ \text{add}(x, y + 1) = \text{add}(x, y) + 1. \]

\[ \text{add}(x, 0) = g(x), \]
where
\[ g: \mathbb{N} \rightarrow \mathbb{N}, \quad g(x) = x, \]
i.e. \( g = \text{id} = \text{proj}_0^1. \)
Addition

\[
\begin{align*}
\text{add}(x, 0) &= x + 0 = g(x), \\
\text{add}(x, y + 1) &= \text{add}(x, y) + 1.
\end{align*}
\]

\[
\text{add}(x, y + 1) = h(x, y, \text{add}(x, y)),
\]

where

\[
h : \mathbb{N}^3 \rightarrow \mathbb{N}, h(x, y, z) := z + 1.
\]

\[
h = \text{succ} \circ \text{proj}_2:
\]

\[
(\text{succ} \circ \text{proj}_2)(x, y, z) = \text{succ}(\text{proj}_2^3(x, y, z))
\]

\[
= \text{succ}(z)
\]

\[
= z + 1
\]

\[
= h(x, y, z).
\]
Addition

\[
\begin{align*}
\text{add}(x, 0) &= x + 0 = g(x), \\
\text{add}(x, y + 1) &= \text{add}(x, y) + 1 = h(x, y, \text{add}(x, y)), \\
g &= \text{proj}^1_0, \\
h &= \text{succ} \circ \text{proj}^3_2.
\end{align*}
\]

Therefore

\[
\text{add} = \text{primrec}(\text{proj}^1_0, \text{succ} \circ \text{proj}^3_2).
\]
Multiplication

\[ \text{mult} : \mathbb{N}^2 \rightarrow \mathbb{N}, \text{mult}(x,y) = x \cdot y \]

is primitive recursive. We have the laws:

\[
\begin{align*}
\text{mult}(x,0) &= x \cdot 0 = 0 \\
\text{mult}(x,y+1) &= x \cdot (y+1) \\
&= x \cdot y + x \\
&= \text{mult}(x,y) + x \\
&= \text{add}(\text{mult}(x,y), x)
\end{align*}
\]

Jump over rest
Multiplication

\[
\text{mult}(x, 0) = 0, \\
\text{mult}(x, y + 1) = \text{add}(\text{mult}(x, y), x).
\]

\[\text{mult}(x, 0) = g(x), \text{ where } g : \mathbb{N} \rightarrow \mathbb{N}, g(x) = 0,\]

i.e. \(g = \text{zero},\)
Multiplication

\[
\begin{align*}
\text{mult}(x, 0) &= 0 = g(x), \\
\text{mult}(x, y + 1) &= \text{add}(\text{mult}(x, y), x).
\end{align*}
\]

\[\text{mult}(x, y + 1) = h(x, y, \text{mult}(x, y)),\]

where
\[h : \mathbb{N}^3 \to \mathbb{N},
\quad h(x, y, z) := \text{add}(z, x).
\]

\[h = \text{add} \circ (\text{proj}_2^3, \text{proj}_0^3):\]

\[
(\text{add} \circ (\text{proj}_2^3, \text{proj}_0^3))(x, y, z) = \text{add}(\text{proj}_2^3(x, y, z), \text{proj}_0^3(x, y, z))
\]

\[
= \text{add}(z, x)
\]

\[
= h(x, y, z).
\]
Multiplication

\[
\begin{align*}
\text{mult}(x, 0) &= 0 = g(x), \\
\text{mult}(x, y + 1) &= \text{add}(\text{mult}(x, y), x) = h(x, y, \text{mult}(x, y)), \\
g &= \text{zero}, \\
h &= \text{add} \circ (\text{proj}_2^3, \text{proj}_0^3).
\end{align*}
\]

Therefore

\[
\text{mult} = \text{primrec}(\text{zero}, \text{add} \circ (\text{proj}_2^3, \text{proj}_0^3)).
\]
Predecessor Function

pred is prim. rec.:

\[ \begin{align*}
\text{pred}(0) & = 0, \\
\text{pred}(x + 1) & = x.
\end{align*} \]
Subtraction

\[ \text{sub}(x, y) = x \div y \] is prim. rec.:

\[
\begin{align*}
\text{sub}(x, 0) &= x, \\
\text{sub}(x, y + 1) &= x \div (y + 1) \\
&= (x \div y) \div 1 \\
&= \text{pred}(\text{sub}(x, y)) .
\end{align*}
\]
Signum Function

\[ \text{sig} : \mathbb{N} \rightarrow \mathbb{N}, \]

\[ \text{sig}(x) := \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x = 0 \end{cases} \]

is prim. rec.:
\[ \text{sig}(x) = x \div (x \div 1): \]

For \( x = 0 \) we have

\[ x \div (x \div 1) = 0 \div (0 \div 1) = 0 \div 0 = 0 = \text{sig}(x). \]

For \( x > 0 \) we have

\[ x \div (x \div 1) = x - (x - 1) = x - x + 1 = 1 = \text{sig}(x). \]
Signum Function

Note that

\[ \text{sig} = \chi_{x>0} \]

where \( x > 0 \) stands for the unary predicate, which is true for \( x \) iff \( x > 0 \):

\[
\chi_{x>0}(y) = \begin{cases} 
1, & \text{if } y > 0, \\
0, & \text{if } y = 0.
\end{cases} = \text{sig}(y)
\]
\( x < y \) is Prim. Rec.

\[ A(x, y) : \iff x < y \text{ is primitive recursive, since} \]
\[ \chi_A(x, y) = \text{sig}(y - x) : \]

1. **If** \( x < y \), **then**

\[ y - x = y - x > 0 , \]

therefore

\[
\text{sig}(y - x) = 1 = \chi_A(x, y)
\]

2. **If** \( \neg(x < y) \), i.e. \( x \geq y \), **then**

\[ y - x = 0 , \]

\[ \text{sig}(y - x) = 0 = \chi_A(x, y) . \]
Consider the sequence of definitions of addition, multiplication, exponentiation:

**Addition:**

\[
\begin{align*}
    n + 0 &= n , \\
    n + (m + 1) &= (n + m) + 1 ,
\end{align*}
\]

Therefore, if we write \((+ 1)\) for the function \(\mathbb{N} \to \mathbb{N}\),
\((+ 1)(n) = n + 1\), then

\[
n + m = ((+ 1)^m(n) .
\]
Remark on Notation

The notation \((+ \ 1)^m(n)\) is to be understood as follows:

Let \(f\) be a function (e.g. \((+ \ 1))\). Then we define

\[
f^n(m) := f\left(f(\cdots f(m) \cdots)\right)\]

This is not to be confused with exponentiation

\[
n^m = \underbrace{n \cdot \cdots \cdot n}_{n \text{ times}}.
\]

So

\[
((+ \ 1)^m(n) = ((+ \ 1)(((+ \ 1)(\cdots ((+ \ 1)(n) \cdots)))
\]

\[
= \underbrace{(\cdots ((m+1) + 1) \cdots + 1)}_{m \text{ times}} = m + n
\]
Multiplication:

\[ n \cdot 0 = 0 , \]
\[ n \cdot (m + 1) = (n \cdot m) + n , \]

Therefore, if we write \((+ n)\) for the function \(\mathbb{N} \to \mathbb{N}\), \((+ n)(k) = k + n\), then

\[ n \cdot m = ((+ n)^m(0) . \]
Add., Mult., Exp.

- **Exponentiation:**

\[
\begin{align*}
    n^0 &= 1, \\
    n^{m+1} &= (n^m) \cdot n,
\end{align*}
\]

Therefore, if we write \(((\cdot) \, n)\) for the function \(\mathbb{N} \to \mathbb{N}\), \(((\cdot) \, n)(m) = n \cdot m\), then

\[
    n^m = (((\cdot) \, n)^m)(1).
\]

- Note that above, we have both occurrences of \(n^m\) for exponentation and of \(((\cdot) \, n)^m(1)\) for iterated function application.
Superexponentiation

Extend this sequence further, by defining

**Superexponentiation:**

\[
\begin{align*}
\text{superexp}(n, 0) &= 1, \\
\text{superexp}(n, m + 1) &= n^{\text{superexp}(n, m)},
\end{align*}
\]

Therefore, if we write \((\uparrow n)\) for the function \(\mathbb{N} \to \mathbb{N}\),

\[
((\uparrow n)(k) = n^k,
\]

then

\[
\text{superexp}(n, m) = ((\uparrow n)^m(1).
\]
Supersuperexponentiation

- **Supersuperexponentiation:**

  \[
  \text{supersuperexp}(n, 0) = 1, \\
  \text{supersuperexp}(n, m + 1) = \text{superexp}(n, \text{supersuperexp}(n, m)),
  \]

- Etc.

- One obtains sequence of extremely fast growing functions.

- These functions will exhaust the primitive recursive functions.

- We will reconsider this sequence at the beginning of Sect. 6 (a).
(b) Closure of the Prim. Rec. Func.

Closure under $\cup$, $\cap$, $\setminus$

- If $R, S \subseteq \mathbb{N}^n$ are prim. rec., so are
  - $R \cup S$,
  - $R \cap S$,
  - $\mathbb{N}^n \setminus R$. 
Closure under Prop. Connectives

Note:

\[(R \cup S)(\vec{x}) \iff R(\vec{x}) \lor S(\vec{x}),\]
\[(R \cap S)(\vec{x}) \iff R(\vec{x}) \land S(\vec{x}),\]
\[(\mathbb{N}^n \setminus R)(\vec{x}) \iff \neg R(\vec{x}).\]

So the prim. rec. predicates are closed under the propositional connectives \(\land, \lor, \neg\).

Example:

Above we have seen that “\(x < y\)” is primitive recursive.

Therefore the predicates “\(x \leq y\)” and “\(x = y\)” are primitive recursive:

\(x \leq y \iff \neg(y < x).\)
\(x = y \iff x \leq y \land y \leq x.\)
Closure under $\cup$, $\cap$, $\setminus$

Proof of $(R \cup S)(\vec{x}) \iff R(\vec{x}) \lor S(\vec{x})$:

$(R \cup S)(\vec{x}) \iff \vec{x} \in R \cup S$
$\iff \vec{x} \in R \lor \vec{x} \in S$
$\iff R(\vec{x}) \lor S(\vec{x})$

Jump over Rest

Proof of $(R \cap S)(\vec{x}) \iff R(\vec{x}) \land S(\vec{x})$:

$(R \cap S)(\vec{x}) \iff \vec{x} \in R \cap S$
$\iff \vec{x} \in R \land \vec{x} \in S$
$\iff R(\vec{x}) \land S(\vec{x})$
Closure under $\cup$, $\cap$, $\setminus$

Proof of $(\mathbb{N}^n \setminus R)(\vec{x}) \iff \neg R(\vec{x})$:

- $(\mathbb{N}^n \setminus R)(\vec{x}) \iff \vec{x} \in (\mathbb{N}^n \setminus R)$
- $\iff \vec{x} \notin R$
- $\iff \neg R(\vec{x})$
Proof of Closure under $\cup$

\[ \chi_{R \cup S}(\vec{x}) = \text{sig}(\chi_R(\vec{x}) + \chi_S(\vec{x})), \]

(therefore $R \cup S$ is primitive recursive):

- If $R(\vec{x})$ holds, then

\[
\text{sig}(\chi_R(\vec{x}) + \chi_S(\vec{x})) = 1 = \chi_{R \cup S}(\vec{x}).
\]

\[
\begin{aligned}
&= 1 \\
&\geq 0 \\
\geq 1 \\
= 1
\end{aligned}
\]
Proof of Closure under $\cup$

Similarly, if $S(\vec{x})$ holds, then

$$\text{sig}(\chi_R(\vec{x}) + \chi_S(\vec{x})) = 1 = \chi_{R \cup S}(\vec{x})$$

$$\begin{align*}
\geq 0 & \quad = 1 \\
\geq 1 & \quad = 1
\end{align*}$$
Proof of Closure under $\bigcup$

If neither $R(\vec{x})$ nor $S(\vec{x})$ holds, then we have

$$\text{sig}(\chi_R(\vec{x}) + \chi_S(\vec{x})) = 0 = \chi_{R \cup S}(\vec{x})$$

$$= 0 = 0 = 0$$
Proof of Closure under $\cap$

$\chi_{R \cap S}(\vec{x}) = \chi_R(\vec{x}) \cdot \chi_S(\vec{x})$

(and therefore $R \cap S$ is primitive recursive):

Jump over Rest of Proof

If $R(\vec{x})$ and $S(\vec{x})$ hold, then

$$\underbrace{\chi_R(\vec{x}) \cdot \chi_S(\vec{x}) = 1 = \chi_{R \cap S}(\vec{x})}.$$
Proof of Closure under $\cap$

- If $\neg R(\vec{x})$ holds, then $\chi_R(\vec{x}) = 0$, therefore
  \[
  \chi_R(\vec{x}) \cdot \chi_S(\vec{x}) = 0 = \chi_{R \cap S}(\vec{x}).
  \]

- Similarly, if $\neg S(\vec{x})$, we have
  \[
  \chi_R(\vec{x}) \cdot \chi_S(\vec{x}) = 0 = \chi_{R \cap S}(\vec{x}).
  \]
Proof of Closure under

\[ \chi_{N^n \setminus R}(\vec{x}) = 1 \div \chi_R(\vec{x}) \]
(and therefore primitive recursive):

**Jump over Rest of Proof**

- If \( R(\vec{x}) \) holds, then \( \chi_R(\vec{x}) = 1 \), therefore

  \[
  1 \div \chi_R(\vec{x}) = 1 = \chi_{N^n \setminus R}(\vec{x}) .
  \]

- If \( R(\vec{x}) \) does not hold, then \( \chi_R(\vec{x}) = 0 \), therefore

  \[
  1 \div \chi_R(\vec{x}) = 1 = \chi_{N^n \setminus R}(\vec{x}) .
  \]
Definition by Cases

The primitive recursive functions are closed under 
**definition by cases:**

Assume

- \( g_1, g_2 : \mathbb{N}^n \to \mathbb{N} \) are primitive recursive,
- \( R \subseteq \mathbb{N}^n \) is primitive recursive.

Then \( f : \mathbb{N}^n \to \mathbb{N} \),

\[
f(\vec{x}) := \begin{cases} 
g_1(\vec{x}), & \text{if } R(\vec{x}), 
g_2(\vec{x}), & \text{if } \neg R(\vec{x}), 
\end{cases}
\]

is primitive recursive.
**Definition by Cases**

\[
f(x) := \begin{cases} 
  g_1(x), & \text{if } R(x), \\
  g_2(x), & \text{if } \neg R(x), 
\end{cases}
\]

\[
f(x) = g_1(x) \cdot \chi_R(x) + g_2(x) \cdot \chi_{\mathbb{N}^n \setminus R}(x)
\]

**Jump over rest of proof.**

- **If** \( R(x) \) **holds, then** \( \chi_R(x) = 1, \)
  \( \chi_{\mathbb{N}^n \setminus R}(x) = 0, \) **therefore**

\[
g_1(x) \cdot \chi_R(x) + g_2(x) \cdot \chi_{\mathbb{N}^n \setminus R}(x) = g_1(x) = f(x).
\]
Definition by Cases

\[ f(\vec{x}) := \begin{cases} 
    g_1(\vec{x}), & \text{if } R(\vec{x}), \\
    g_2(\vec{x}), & \text{if } \neg R(\vec{x}),
\end{cases} \]

Show

\[ f(\vec{x}) = g_1(\vec{x}) \cdot \chi_R(\vec{x}) + g_2(\vec{x}) \cdot \chi_{\mathbb{N}^n \setminus R}(\vec{x}) : \]

If \( \neg R(\vec{x}) \) holds, then \( \chi_R(\vec{x}) = 0 \), \( \chi_{\mathbb{N}^n \setminus R}(\vec{x}) = 1 \),

\[ g_1(\vec{x}) \cdot \chi_R(\vec{x}) + g_2(\vec{x}) \cdot \chi_{\mathbb{N}^n \setminus R}(\vec{x}) = g_2(\vec{x}) = f(\vec{x}) . \]
Bounded Sums

If \( g : \mathbb{N}^{n+1} \to \mathbb{N} \) is prim. rec., so is

\[
f : \mathbb{N}^{n+1} \to \mathbb{N} , \quad f(\vec{x}, y) := \sum_{z<y} g(\vec{x}, z) ,
\]

where

\[
\sum_{z<0} g(\vec{x}, z) := 0 ,
\]

and for \( y > 0 \),

\[
\sum_{z<y} g(\vec{x}, z) := g(\vec{x}, 0) + g(\vec{x}, 1) + \cdots + g(\vec{x}, y - 1) .
\]
Bounded Sums

\[ f : \mathbb{N}^{n+1} \rightarrow \mathbb{N} , \quad f(\vec{x}, y) := \sum_{z<y} g(\vec{x}, z) \]

Proof that \( f \) is prim. rec.:

\[
\begin{align*}
f(\vec{x}, 0) & = 0 , \\
f(\vec{x}, y + 1) & = f(\vec{x}, y) + g(\vec{x}, y) .
\end{align*}
\]

Jump over rest of proof. The last equations follows from

\[
\begin{align*}
f(\vec{x}, y + 1) & = \sum_{z<y+1} g(\vec{x}, z) \\
& = (\sum_{z<y} g(\vec{x}, z)) + g(\vec{x}, y) \\
& = f(\vec{x}, y) + g(\vec{x}, y) .
\end{align*}
\]
Example

- We have above

\[ f(\vec{x}, 0) = g(\vec{x}, 0) \]
\[ f(\vec{x}, 1) = g(\vec{x}, 0) + g(\vec{x}, 1) \]
\[ = f(\vec{x}, 0) + g(\vec{x}, 0) \]
\[ f(\vec{x}, 2) = g(\vec{x}, 0) + g(\vec{x}, 1) + g(\vec{x}, 2) \]
\[ = f(\vec{x}, 1) + g(\vec{x}, 2) \]

etc.
Bounded Products

If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is prim. rec., so is

$$f : \mathbb{N}^{n+1} \to \mathbb{N}, \quad f(\vec{x}, y) := \prod_{z<y} g(\vec{x}, z),$$

where

$$\prod_{z<0} g(\vec{x}, z) := 1,$$

and for $y > 0$,

$$\prod_{z<y} g(\vec{x}, z) := g(\vec{x}, 0) \cdot g(\vec{x}, 1) \cdots g(\vec{x}, y-1).$$

Omit Proof
Bounded Products

\[ f : \mathbb{N}^{n+1} \to \mathbb{N} \ , \quad f(\vec{x}, y) := \prod_{z<y} g(\vec{x}, z) \ , \]

Proof that \( f \) is prim. rec.:

\[
\begin{align*}
    f(\vec{x}, 0) & = 1 \\
    f(\vec{x}, y + 1) & = f(\vec{x}, y) \cdot g(\vec{x}, y) .
\end{align*}
\]

Here, the last equations follows by

\[
\begin{align*}
    f(\vec{x}, y + 1) & = \prod_{z<y+1} g(\vec{x}, z) \\
                    & = (\prod_{z<y} g(\vec{x}, z)) \cdot g(\vec{x}, y) \\
                    & = f(\vec{x}, y) \cdot g(\vec{x}, y) .
\end{align*}
\]
Example

Example for closure under bounded products:

\[ f : \mathbb{N} \rightarrow \mathbb{N}, \]

\[ f(n) := n! = 1 \cdot 2 \cdot \ldots \cdot n \]

\((f(0) = 0! = 1),\)

is primitive recursive, since

\[ f(n) = \prod_{i<n}(i + 1) = \prod_{i<n}g(i), \]

where \(g(i) := i + 1\) is prim. rec..

(Note that in the special case \(n = 0\) we have

\[ f(0) = 0! = 1 = \prod_{i<0}(i + 1). \]
Alternatively, the factorial function can be defined directly by using primitive recursion as follows:

\[
\begin{align*}
0! &= 1 \\
(n + 1)! &= n! \cdot (n + 1)
\end{align*}
\]
Bounded Quantification

If $R \subseteq \mathbb{N}^{n+1}$ is prim. rec., so are

$$R_1(\vec{x}, y) :\iff \forall z < y. R(\vec{x}, z) ,$$

$$R_2(\vec{x}, y) :\iff \exists z < y. R(\vec{x}, z) .$$
Bounded Quantification

\[ R_1(\vec{x}, y) :\iff \forall z < y. R(\vec{x}, z) , \]

Proof for \( R_1 \):

\[ \chi_{R_1}(\vec{x}, y) = \prod_{z < y} \chi_{R}(\vec{x}, z) : \]

Jump over details.

- If \( \forall z < y. R(\vec{x}, z) \) holds,
  then \( \forall z < y. \chi_{R}(\vec{x}, z) = 1 \),
  therefore

\[ \prod_{z < y} \chi_{R}(\vec{x}, y) = \prod_{z < y} 1 = 1 = \chi_{R_1}(\vec{x}, y) . \]
Bounded Quantification

\[ R_1(\vec{x}, y) \iff \forall z < y. R(\vec{x}, z) \]

Show \( \chi_{R_1}(\vec{x}, y) = \prod_{z<y} \chi_{R}(\vec{x}, z) \).

If \( \neg R(\vec{x}, z) \) for one \( z < y \), then \( \chi_{R}(\vec{x}, z) = 0 \), therefore

\[ \prod_{z<y} \chi_{R}(\vec{x}, y) = 0 = \chi_{R_1}(\vec{x}, y) \]
Bounded Quantification

\[ R_2(\vec{x}, y) :\iff \exists z < y. R(\vec{x}, z) \]

**Proof for** \( R_2 \):

\[ \chi_{R_2}(\vec{x}, y) = \mathsf{sig}\left( \sum_{z < y} \chi_{R}(\vec{x}, z) \right) : \]

Jump over Rest of Proof
- If \( \forall z < y. \neg R(\vec{x}, z) \), then

\[
\mathsf{sig}\left( \sum_{z < y} \chi_{R}(\vec{x}, y) \right) = \mathsf{sig}\left( \sum_{z < y} 0 \right) \\
= \mathsf{sig}(0) \\
= 0 \\
= \chi_{R_2}(\vec{x}, y). 
\]
Bounded Quantification

\[ R_2(\vec{x}, y) :\iff \exists z < y. R(\vec{x}, z) . \]

Show \( \chi_{R_2}(\vec{x}, y) = \text{sig}(\sum_{z<y} \chi_{R}(\vec{x}, z)) \)

- If \( R(\vec{x}, z) \), for some \( z < y \), then
  \( \chi_{R}(\vec{x}, z) = 1 \), therefore

  \[
  \sum_{z<y} \chi_{R}(\vec{x}, y) \geq \chi_{R}(\vec{x}, z) = 1 ,
  \]

  therefore

  \[
  \text{sig}(\sum_{z<y} \chi_{R}(\vec{x}, y)) = 1 = \chi_{R_2}(\vec{x}, y) .
  \]
Bounded Search

If $R \subseteq \mathbb{N}^{n+1}$ is a prim. rec. predicate, so is

$$f(\bar{x}, y) := \mu z < y. R(\bar{x}, z),$$

where

$$\mu z < y. R(\bar{x}, z) := \begin{cases} \text{the least } z \text{ s.t. } R(\bar{x}, z) \text{ holds,} & \text{if such } z \text{ exists,} \\ y & \text{otherwise.} \end{cases}$$
Bounded Search

\[ f(\vec{x}, y) := \mu z < y. R(\vec{x}, z) \]

**Proof:**

Define

\[ Q(\vec{x}, y) :\Leftrightarrow R(\vec{x}, y) \land \forall z < y. \neg R(\vec{x}, z) , \]

\[ Q'(\vec{x}, y) :\Leftrightarrow \forall z < y. \neg R(\vec{x}, z) \]

\( Q \) and \( Q' \) are primitive recursive.

\( Q(\vec{x}, y) \) holds, if \( y \) is minimal s.t. \( R(\vec{x}, y) \).

We show

\[ f(\vec{x}, y) = (\sum_{z<y} \chi Q(\vec{x}, z) \cdot z) + \chi Q'(\vec{x}, y) \cdot y . \]

Jump over details.
Bounded Search

\[ Q(\vec{x}, y) :\iff R(\vec{x}, y) \land \forall z < y. \neg R(\vec{x}, z), \]
\[ Q'(\vec{x}, y) :\iff \forall z < y. \neg R(\vec{x}, z), \]

Show \[ f(\vec{x}, y) = \left( \sum_{z < y} \chi_{Q}(\vec{x}, z) \cdot z \right) + \chi_{Q'}(\vec{x}, y) \cdot y. \]

Assume \( \exists z < y. R(\vec{x}, z). \)
Let \( z \) be minimal s.t. \( R(\vec{x}, z). \)
\[ \Rightarrow Q(\vec{x}, z), \]
\[ \Rightarrow \chi_{Q}(\vec{x}, z) \cdot z = z. \]
For \( z \neq z' \) we have \( \neg Q(\vec{x}, z'), \)
therefore \[ \chi_{Q}(\vec{x}, z') \cdot z' = 0 \ (z' \neq z). \]
Furthermore, \( \neg Q'(\vec{x}, y), \) therefore \[ \chi_{Q'}(\vec{x}, y) \cdot y = 0. \]
Therefore
\[ \left( \sum_{z < y} \chi_{Q}(\vec{x}, z) \cdot z \right) + \chi_{Q'}(\vec{x}, y) \cdot y = z = \mu z' < y.R(\vec{x}, z'). \]
Bounded Search

\[ Q(\vec{x}, y) :\Leftrightarrow R(\vec{x}, y) \land \forall z < y. \neg R(\vec{x}, z) , \]
\[ Q'(\vec{x}, y) :\Leftrightarrow \forall z < y. \neg R(\vec{x}, z) , \]

Show \( f(\vec{x}, y) = (\sum_{z<y} \chi Q(\vec{x}, z) \cdot z) + \chi Q'(\vec{x}, y) \cdot y . \)

Assume \( \forall z < y. \neg R(\vec{x}, z) . \)
\[ \Rightarrow \neg Q(\vec{x}, z) \text{ for } z < y , \]
\[ \Rightarrow \forall z < y. \chi Q(\vec{x}, z) \cdot z = 0 . \]

Furthermore, \( Q'(\vec{x}, y) , \)
then \( \chi Q'(\vec{x}, y) \cdot y = y . \)

Therefore

\[ (\sum_{z<y} \chi Q(\vec{x}, z) \cdot z) + \chi Q'(\vec{x}, y) \cdot y = y = \mu z' < y. R(\vec{x}, z') . \]
Bounded Search

\[ f(\vec{x}, y) := \mu z < y. R(\vec{x}, z) \]

Alternatively, \( f \) can be defined by primitive recursion directly using the equations:

\[
\begin{align*}
  f(\vec{x}, 0) &= 0 \\
  f(\vec{x}, y + 1) &= \begin{cases} 
    f(\vec{x}, y) & \text{if } f(\vec{x}, y) < y, \\
    y & \text{if } f(\vec{x}, y) = y \land R(\vec{x}, y), \\
    y + 1 & \text{otherwise.}
  \end{cases}
\end{align*}
\]

Exercise: Show

- \( f \) fulfills those equations
- From these equations it follows that \( f \) is primitive recursive, provided \( R \) is.
Example

Let $P \subseteq \mathbb{N}$ be a primitive recursive predicate, and define

$$f : \mathbb{N} \rightarrow \mathbb{N},$$

$$f(x) := |\{y < x \mid P(y)\}| .$$

$f(x)$ is the number of $y < x$ s.t. $P(y)$ holds. $f$ is primitive recursive, since

$$f(x) = \sum_{y < x} \chi_{P(y)} .$$
Example 2

Let $Q \subseteq \mathbb{N}$ be a primitive recursive predicate.

We show how to determine primitive recursively the second least $y < x$ s.t. $Q(y)$ holds.

**Step1:** Express the property to be the second least $y < x$ s.t. $Q(y)$ holds as a prim. rec. predicate $P(y)$:

$$P(y) : \iff Q(y) \land (\exists z < y. Q(z)) \land \neg(\exists z < y. \exists z' < y. (Q(z) \land Q(z') \land z \neq z'))$$

$P(y)$ is primitive recursive, since it is defined from $Q$ using $\land$, $\neg$, bounded quantification and “$z = z'$”.
Example 2

Step 2: Let \( f(y) \) be the second least \( y < x \) s.t. \( Q(y) \) holds:

\[
f(x) = \begin{cases} 
  y, & \text{if } y < x \text{ and } P(y), \\
  x, & \text{if there is no } y < x \text{ s.t. } P(y).
\end{cases}
\]

Then

\[
f(x) = \mu y < x. P(y)
\]

so \( f \) is primitive recursive.

(We could have defined instead

\[
P'(y) :\iff Q(y) \land \exists z < y. Q(z)
\]

Then \( f(x) = \mu y < x. P'(y) \) holds.)
Lemma 5.1

The following functions are primitive recursive:

(a) $\pi : \mathbb{N}^2 \rightarrow \mathbb{N}$.
   (Remember, $\pi(n, m)$ encodes two natural numbers as one.)

(b) $\pi_0, \pi_1 : \mathbb{N} \rightarrow \mathbb{N}$.
   (Remember $\pi_0(\pi(n, m)) = n$, $\pi_1(\pi(n, m)) = m$).

(c) $\pi^k : \mathbb{N}^k \rightarrow \mathbb{N}$ ($k \geq 1$).
   (Remember $\pi^k(n_0, \ldots, n_{k-1})$ encodes the sequence $(n_0, \ldots, n_k)$.)
Lemma 5.1

(d) \( f : \mathbb{N}^3 \to \mathbb{N}, \)

\[
f(x, k, i) = \begin{cases} 
\pi^k_i(x), & \text{if } i < k, \\
x, & \text{otherwise.}
\end{cases}
\]

(Remember that \( \pi^k_i(\pi^k(n_0, \ldots, n_{k-1})) = n_i \) for \( i < k \).)

We write \( \pi^k_i(a) \) for \( f(x, k, i) \), even if \( i \geq k \).

(e) \( f_k : \mathbb{N}^k \to \mathbb{N}, \)

\[
f_k(x_0, \ldots, x_{k-1}) = \langle x_0, \ldots, x_{k-1} \rangle.
\]

(Remember that \( \langle x_0, \ldots, x_{k-1} \rangle \) encodes the sequence \( x_0, \ldots, x_{k-1} \) as one natural number.

(f) \( \text{lh} : \mathbb{N} \to \mathbb{N}. \)

(Remember that \( \text{lh}(\langle x_0, \ldots, x_{k-1} \rangle) = k \).)
Lemma 5.1

\(g : \mathbb{N}^2 \rightarrow \mathbb{N}, \, g(x, i) = (x)_i.\)

(Remember that \((\langle x_0, \ldots, x_{k-1} \rangle)_i = x_i\) for \(i < k.\))

The proof will be omitted in the lecture.

Jump over proof.
Proof of Lemma 5.1 (a), (b)

(a) \[
\pi(n, m) = \left( \sum_{i \leq n+m} i \right) + m
\]
\[
= \left( \sum_{i < n+m+1} i \right) + m
\]
is primitive recursive.

(b) One can easily show that \( n, m \leq \pi(n, m) \).
Therefore we can define
\[
\pi_0(n) := \mu k < n + 1. \exists l < n + 1. n = \pi(k, l),
\]
\[
\pi_1(n) := \mu l < n + 1. \exists k < n + 1. n = \pi(k, l).
\]
Therefore \( \pi_0, \pi_1 \) are primitive recursive.
Proof of Lemma 5.1 (c)

(c) Proof by induction on $k$:

- $k = 1$: $\pi^1(x) = x$, so $\pi^1$ is primitive recursive.

- $k \rightarrow k + 1$: Assume that $\pi^k$ is primitive recursive. Show that $\pi^{k+1}$ is primitive recursive as well:

$$\pi^{k+1}(x_0, \ldots, x_k) = \pi(\pi^k(x_0, \ldots, x_{k-1}), x_k).$$

Therefore $\pi^{k+1}$ is primitive recursive (using that $\pi$, $\pi^k$ are primitive recursive).
Proof of Lemma 5.1 (d)

(d) We have

\begin{align*}
\pi_0^1(x) & = x, \\
\pi_i^{k+1}(x) & = \pi_i^k(\pi_0(x)), \text{ if } i < k, \\
\pi_i^{k+1}(x) & = \pi_1(x), \text{ if } i = k,
\end{align*}

Therefore

\[ \pi_i^k(x) = \begin{cases} 
\pi_1((\pi_0)^{k-i}(x)), & \text{if } i > 0, \\
(\pi_0)^k(x), & \text{if } i = 0.
\end{cases} \]
Proof of Lemma 5.1 (d)

and

\[ f(x, k, i) = \begin{cases} 
  x, & \text{if } i \geq k, \\
  \pi_1((\pi_0)^{k-i}(x)), & \text{if } 0 < i < k, \\
  (\pi_0)^k(x), & \text{if } i = 0 < k.
\end{cases} \]

Define \( g : \mathbb{N}^2 \to \mathbb{N}, \)

\[
\begin{align*}
g(x, 0) & := x, \\
g(x, k + 1) & := \pi_0(g(x, k)),
\end{align*}
\]

which is primitive recursive.
Proof of Lemma 5.1 (d)

Then we get \( g(x, k) = (\pi_0)^k(x) \), therefore

\[
 f(x, k, i) = \begin{cases} 
 x, & \text{if } i \geq k, \\
 \pi_1(g(x, k - i)), & \text{if } 0 < i < k, \\
 g(x, k), & \text{if } i = 0 < k. 
\end{cases}
\]

So \( f \) is primitive recursive.
Proof of Lemma 5.1 (e), (f), (g)

(e) \[ f_k(x_0, \ldots, x_{k-1}) = 1 + \pi(k - 1, \pi^k(x_0, \ldots, x_{k-1})) \]
is primitive recursive.

(f) \[ lh(x) = \begin{cases} 
0, & \text{if } x = 0, \\
\pi_0(x - 1) + 1, & \text{if } x \neq 0.
\end{cases} \]

(g) \[ (x)_i = \pi^\text{lh}(x)_i(\pi_1(x - 1)) \]
\[ = f(\pi_1(x - 1), lh(x), i) \]
is primitive recursive.
Lemma and Definition 5.2

Prim. rec. functions as follows do exist:

(a) \( \text{snoc} : \mathbb{N}^2 \rightarrow \mathbb{N} \) s.t.

\[
\text{snoc}(\langle x_0, \ldots, x_{n-1} \rangle, x) = \langle x_0, \ldots, x_{n-1}, x \rangle.
\]

**Remark:** \( \text{snoc} \) is the word \( \text{cons} \) reversed.
\( \text{snoc} \) is like \( \text{cons} \), but adds an element to the end rather than to the beginning of a list.

(b) \( \text{last} : \mathbb{N} \rightarrow \mathbb{N} \) and \( \text{beginning} : \mathbb{N} \rightarrow \mathbb{N} \) s.t.

\[
\text{last}(%(\text{snoc}(x, y)) \quad = \quad y,
\text{beginning}(\text{snoc}(x, y)) \quad = \quad x.
\]

The proof will be omitted in the lecture.

Jump over proof.
Proof of Lemma 5.2 (a)

Define

\[ \text{snoc}(x, y) = \begin{cases} 
\langle y \rangle, & \text{if } x = 0, \\
1 + \pi(lh(x), \pi(\pi_1(x - 1), y)), & \text{otherwise,}
\end{cases} \]

so \text{snoc} is primitive recursive.
Proof of Lemma 5.2 (a)

We have

\[
\text{snoc}(\langle \rangle, y) = \text{snoc}(0, y) = \langle y \rangle, \\
\text{snoc}(\langle x_0, \ldots, x_k \rangle, y) = \text{snoc}(1 + \pi(k, \pi^{k+1}(x_0, \ldots, x_k)), y) = 1 + \pi(k + 1, \pi(\pi_1((1 + \pi(k, \pi^{k+1}(x_0, \ldots, x_k))) \div 1), y))
\]

(by \(\text{lh}(\langle x_0, \ldots, x_k \rangle) = k + 1\))

\[
= 1 + \pi(k + 1, \pi(\pi^{k+1}(x_0, \ldots, x_k), y)) = 1 + \pi(k + 1, \pi^{k+2}(x_0, \ldots, x_k, y)) = \langle x_0, \ldots, x_k, y \rangle.
\]
Proof of Lemma 5.2 (b)

**Proof for beginning:**
Define

\[
\text{beginning}(x) := \begin{cases} 
\langle \rangle, & \text{if } \lh(x) \leq 1, \\
\langle (x)_0 \rangle, & \text{if } \lh(x) = 2, \\
1 + \pi((\lh(x) \div 1) \div 1, \pi_0(\pi_1(y \div 1))), & \text{otherwise.}
\end{cases}
\]
Proof of Lemma 5.2 (b)

Let $x = \text{snoc}(y, z)$. Show $\text{beginning}(x) = y$.

**Case** $\text{lh}(y) = 0$: Then

$$x = \text{snoc}(y, z) = \langle z \rangle$$

therefore $\text{lh}(x) = 1$, and

$$\text{beginning}(x) = \langle \rangle$$

$$= y$$
Proof of Lemma 5.2 (b)

Case \( \text{lh}(y) = 1 \): Then \( y = \langle y' \rangle \) for some \( y' \), \( \text{snoc}(y, z) = \langle y', z \rangle \),

\[
\text{beginning}(x) = \langle (x)_0 \rangle \\
= \langle (\langle y', z \rangle)_0 \rangle \\
= \langle y' \rangle \\
= y
\]
Proof of Lemma 5.2 (b)

Case \( \text{lh}(y) > 1 \): Let \( \text{lh}(y) = n + 2 \),

\[
y = \langle y_0, \ldots, y_{n+1} \rangle = 1 + \pi(n + 1, \pi^{n+2}(y_0, \ldots, y_{n+1})) .
\]

Then

\[
\text{snoc}(y, z) = 1 + \pi(n + 2, \pi(\pi_1(y - 1), z)) .
\]
Proof of Lemma 5.2 (b)

Therefore

\[
\begin{align*}
\text{beginning}(\text{snoc}(y, z)) &= 1 + \pi(((\text{lh}(x) - 1) - 1), \pi_0(\pi_1(\text{snoc}(y, z) - 1))) \\
&= 1 + \pi(n, \pi_0(\pi_1((1 + \pi(n + 2, \pi(\pi_1(y - 1), z)) - 1)))) \\
&= 1 + \pi(n, \pi_0(\pi(\pi_1(y - 1), z)))) \\
&= 1 + \pi(n, \pi_1(y - 1)) \\
&= 1 + \pi(n, \pi_1((1 + \pi(n + 1, \pi^{n+2}(y_0, \ldots, y_{n+1}))) - 1)) \\
&= 1 + \pi(n, \pi_1(\pi(n + 1, \pi^{n+2}(y_0, \ldots, y_{n+1})))) \\
&= 1 + \pi(n, \pi^{n+2}(y_0, \ldots, y_{n+1})) \\
&= y.
\end{align*}
\]
Proof of Lemma 5.2 (b)

Proof for last:
Define

\[ \text{last}(x) := (x)_{\text{lh}(x)} - 1 \]

If \( y = \langle y_0, \ldots, y_{n-1} \rangle \), then

\[
\text{last}(\text{snoc}(y, z)) = \text{last}(\langle y_0, \ldots, y_{n-1}, z \rangle)
\]

\[
= (\langle y_0, \ldots, y_{n-1}, z \rangle)_{\text{lh}(\langle y_0, \ldots, y_{n-1}, z \rangle)} - 1
\]

\[
= (\langle y_0, \ldots, y_{n-1}, z \rangle)_{n}
\]

\[
= z.
\]
Definition Course-Of-Value

Assume \( f : \mathbb{N}^{n+1} \to \mathbb{N} \). Then we define

\[
\overline{f} : \mathbb{N}^{n+1} \to \mathbb{N} \\
\overline{f}(\vec{x}, n) := \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, n-1) \rangle
\]

Especially \( \overline{f}(\vec{x}, 0) = \langle \rangle \).

\( \overline{f} \) is called the course-of-value function associated with \( f \).
Course-of-Value Prim. Recursion

The prim. rec. functions are closed under course-of-value primitive recursion:

Assume

\[ g : \mathbb{N}^{n+2} \rightarrow \mathbb{N} \]

is primitive recursive. Then

\[ f : \mathbb{N}^{n+1} \rightarrow \mathbb{N} \]

\[ f(\bar{x}, k) = g(\bar{x}, k, f(\bar{x}, k)) \]

is prim. rec.
Course-of-Value Prim. Recursion

**Informal meaning** of course-of-value primitive recursion: If we can express $f(\bar{x}, y)$ by an expression using

- constants,
- $\bar{x}, y$,
- previously defined prim. rec. functions,
- $f(\bar{x}, z)$ for $z < y$,

then $f$ is prim. rec.
Example

Fibonacci numbers are prim. rec.

\( \text{fib} : \mathbb{N} \rightarrow \mathbb{N} \) given by:

\[
\begin{align*}
\text{fib}(0) & := 1, \\
\text{fib}(1) & := 1, \\
\text{fib}(n) & := \text{fib}(n - 1) + \text{fib}(n - 2), \text{ if } n > 1,
\end{align*}
\]

Definable by course-of-value primitive recursion:

\[\text{We have}
\begin{align*}
\text{fib}(n) &= \begin{cases} 
1 & \text{if } n \leq 1, \\
(\text{fib}(n))_{n-2} + (\text{fib}(n))_{n-1} & \text{otherwise.}
\end{cases}
\end{align*}\]
Proof

Proof that prim. rec. functions are closed under course-of-value primitive recursion:
Let $f$ be defined by

$$f(\vec{x}, y) = g(\vec{x}, y, \overline{f}(\vec{x}, y))$$

Show $f$ is prim. rec.

We show first that $\overline{f}$ is primitive recursive.
Proof

\[ f(\vec{x}, y) = g(\vec{x}, y, \overline{f}(\vec{x}, y)) \]

\[ \overline{f}(\vec{x}, 0) = \langle \rangle, \]
\[ \overline{f}(\vec{x}, y + 1) = \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y - 1), f(\vec{x}, y) \rangle \]
\[ = \text{snoc}(\langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y - 1) \rangle, f(\vec{x}, y)) \]
\[ = \text{snoc}(\overline{f}(\vec{x}, y), f(\vec{x}, y)) \]
\[ = \text{snoc}(\overline{f}(\vec{x}, y), g(\vec{x}, y, \overline{f}(\vec{x}, y))) \] .

Therefore \( \overline{f} \) is primitive recursive.
Proof

\[ f(\vec{x}, y) = g(\vec{x}, y, \overline{f}(\vec{x}, y)) \]

Now we have that

\[ f(\vec{x}, y) = (\langle f(\vec{x}, 0), \ldots, f(\vec{x}, y) \rangle)_y \]
\[ = (\overline{f}(\vec{x}, y + 1))_y \]
\[ = \text{last}(\overline{f}(\vec{x}, y + 1)) \]

is primitive recursive.
Lemma and Definition 5.3

There exists prim. rec. functions as follows:

(a) append : \( \mathbb{N}^2 \to \mathbb{N} \) s.t.

\[
\text{append}(\langle n_0, \ldots, n_{k-1} \rangle, \langle m_0, \ldots, m_{l-1} \rangle) = \langle n_0, \ldots, n_{k-1}, m_0, \ldots, m_{l-1} \rangle .
\]

We write \( n \ast m \) for \( \text{append}(n, m) \).

(b) subst : \( \mathbb{N}^3 \to \mathbb{N} \), s.t. if \( i < n \) then

\[
\text{subst}(\langle x_0, \ldots, x_{n-1} \rangle, i, y) = \langle x_0, \ldots, x_{i-1}, y, x_{i+1}, x_{i+2}, \ldots, x_{n-1} \rangle ,
\]

and if \( i \geq n \), then

\[
\text{subst}(\langle x_0, \ldots, x_{n-1} \rangle, i, y) = \langle x_0, \ldots, x_{n-1} \rangle .
\]

We write \( x[i/y] \) for \( \text{subst}(x, i, y) \).
Lemma and Definition 5.3

(c) \( \text{subseq} : \mathbb{N}^3 \rightarrow \mathbb{N} \) s.t., if \( i < n \),

\[ \text{subseq}(\langle x_0, \ldots, x_{n-1}\rangle, i, j) = \langle x_i, x_{i+1}, \ldots, x_{\min(j-1,n-1)}\rangle , \]

and if \( i \geq n \),

\[ \text{subseq}(\langle x_0, \ldots, x_{n-1}\rangle, i, j) = \langle \rangle . \]
(d) \( \text{half} : \mathbb{N} \rightarrow \mathbb{N}, \)  
\text{s.t.} \( \text{half}(n) = k \) if \( n = 2k \) or \( n = 2k + 1 \).

(e) The function \( \text{bin} : \mathbb{N} \rightarrow \mathbb{N}, \) s.t. \( \text{bin}(n) = \langle b_0, \ldots, b_k \rangle, \) for \( b_i \) in normal form (no leading zeros, unless \( n = 0 \)), s.t. \( n = (b_0, \ldots, b_k)_2 \)

(f) A function \( \text{bin}^{-1} : \mathbb{N} \rightarrow \mathbb{N}, \) s.t. \( \text{bin}^{-1}(\langle b_0, \ldots, b_k \rangle) = n, \) if \( (b_0, \ldots, b_k)_2 = n. \)

The proof will be omitted in the lecture.

Jump over proof.
Proof of Lemma 5.3 (a)

We have

\[ \text{append}(\langle x_0, \ldots, x_n \rangle, 0) \]
\[ = \text{append}(\langle x_0, \ldots, x_n \rangle, \langle \rangle) \]
\[ = \langle x_0, \ldots, x_n \rangle, \]

and for \( m > 0 \)

\[ \text{append}(\langle x_0, \ldots, x_n \rangle, \langle y_0, \ldots, y_m \rangle) \]
\[ = \langle x_0, \ldots, x_n, y_0, \ldots, y_m \rangle \]
\[ = \text{sno}c(\langle x_0, \ldots, x_n, y_0, \ldots, y_{m-1} \rangle, y_m) \]
\[ = \text{sno}c(\text{append}(\langle x_0, \ldots, x_n \rangle, \langle y_0, \ldots, y_{m-1} \rangle), y_m) \]
\[ = \text{sno}c(\text{append}(\langle x_0, \ldots, x_n \rangle, \text{beginning}(\langle y_0, \ldots, y_m \rangle)), \text{last}(\langle y_0, \ldots, y_m \rangle)) \].
Proof of Lemma 5.3 (a)

Therefore we have

\[
\text{append}(x, 0) = x,
\]
\[
\text{append}(x, y) = \text{snoc}(\text{append}(x, \text{beginning}(y)), \text{last}(y)),
\]

One can see that \(\text{beginning}(x) < x\) for \(x > 0\), therefore the last equations give a definition of \(\text{append}\) by course-of-value primitive recursion, therefore \(\text{append}\) is primitive recursive.
Proof of Lemma 5.3 (b)

We have

\[
\text{subst}(x, i, y) := \begin{cases} 
    x, & \text{if } \text{lh}(x) \leq i, \\
    \text{snoc} (\text{beginning}(x), y), & \text{if } i + 1 = \text{lh}(x), \\
    \text{snoc} (\text{subst} (\text{beginning}(x), i, y), \text{last}(x)), & \text{if } i + 1 < \text{lh}(x).
\end{cases}
\]

Therefore \text{subst} is definable by course-of-value primitive recursion.
Proof of Lemma 5.3 (c)

We can define

\[
\text{subseq}(x, i, j) = \begin{cases} 
\langle \rangle, & \text{if } i \geq \text{lh}(x), \\
\text{subseq}(\text{beginning}(x), i, j), & \text{if } i < \text{lh}(x) \\
\text{snoc}(	ext{subseq}(\text{beginning}(x), i, j), \text{last}(x)) & \text{if } i < \text{lh}(x) \leq j,
\end{cases}
\]

which is a definition by course-of-value primitive recursion.
Proof of Lemma 5.3 (d), (e)

(d) \( \text{half}(x) = \mu y < x. (2 \cdot y = x \lor 2 \cdot y + 1 = x) \).

(e) \[
\begin{align*}
\text{bin}(x) = \left\{ \begin{array}{ll}
\langle 0 \rangle, & \text{if } x = 0, \\
\langle 1 \rangle & \text{if } x = 1, \\
\text{snoc(\text{half}(x), x \div (2 \cdot \text{half}(x)))}, & \text{if } x > 1.
\end{array} \right.
\end{align*}
\]

therefore definable by course-of-value primitive recursion.
Proof of Lemma 5.3 (f)

\[
\text{bin}^{-1}(x) = \begin{cases} 
0, & \text{if } \text{lh}(x) = 0, \\
(x)_0 & \text{if } \text{lh}(x) = 1, \\
\text{bin}^{-1}(\text{beginning}(x)) \cdot 2 + \text{last}(x) & \text{if } \text{lh}(x) > 1,
\end{cases}
\]

therefore definable by course-of-value primitive recursion.