5. Algebraic View of Computability

- URM and TM based on universal programming languages.
- Next model: Functions generated from
  - basic functions
  - by using certain operations.
- First proposed by Gödel and Kleene 1936.
- Best model for showing that functions **are computable.**

Overview

(c) Introduction of the **partial recursive functions**.
- Third model of computation.
- Partial rec. functions is the main model used for proving theorems in computability theory.

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 primitive recursive functions is a reach set of functions, closed under many operations.

(a) Introd. of the Prim.-Rec. Funct

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

  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
  - $k \in \mathbb{N}$,
  - $h : \mathbb{N}^2 \to \mathbb{N}$ is primitive recursive,

  so is the function $f : \mathbb{N} \to \mathbb{N}$, defined by primitive recursion from k and h.
- Remember that $f := \mathsf{primrec}(k, h)$ is defined by
  - $f(0) = k$,
  - $f(n+1) = h(n, f(n))$.
- f is denoted by $\mathsf{primrec}(k, h)$.

Def. Prim.-Rec. Functions

- If
  - $g : \mathbb{N}^n \to \mathbb{N}$,
  - $h : \mathbb{N}^{n+2} \to \mathbb{N}$ are primitive recursive,

  so is the function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ 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 $\mathsf{primrec}(g, h)$.

Primitive-Rec. Relations

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

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

is primitive recursive.

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 $\_ \circ (\_, \ldots, \_)$
    - i.e. by forming from f, g_i $f \circ (g_0, \ldots, g_{n-1})$
  - and primrec.

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).

Inductively Defined Sets

E.g.

- $\underbrace{\mathsf{primrec}(\underbrace{\mathsf{proj}_0^1}_{:\mathbb{N}\to\mathbb{N}}, \underbrace{\mathsf{succ} \circ \underbrace{\mathsf{proj}_2^3}_{:\mathbb{N}^3\to\mathbb{N}}}_{:\mathbb{N}^3\to\mathbb{N}})}_{:\mathbb{N}^2\to\mathbb{N}} : \mathbb{N}^2 \to \mathbb{N}$ is prim. rec.

  (= addition)
- $\underbrace{\mathsf{primrec}(\underbrace{0}_{\in\mathbb{N}}, \underbrace{\mathsf{proj}_0^2}_{:\mathbb{N}^2\to\mathbb{N}})}_{:\mathbb{N}^1\to\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ is prim. rec.

  (= pred)

Remark

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

Constant Function

- $\mathsf{const}_n : \mathbb{N} \to \mathbb{N}$, $\mathsf{const}_n(k) = n$ is primitive recursive:
  $\mathsf{const}_n = \underbrace{\mathsf{succ} \circ \cdots \circ \mathsf{succ}}_{n \text{ times}} \circ \mathsf{zero}$:

$$\begin{aligned}
\underbrace{\mathsf{succ} \circ \cdots \circ \mathsf{succ}}_{n \text{ times}} \circ \mathsf{zero}(k) &= \underbrace{\mathsf{succ}(\mathsf{succ}(\cdots \mathsf{succ}}_{n \text{ times}}(\mathsf{zero}(k)))) \\
&= \underbrace{\mathsf{succ}(\mathsf{succ}(\cdots \mathsf{succ}}_{n \text{ times}}(0))) \\
&= 0 \underbrace{+1+1\cdots+1}_{n \text{ times}} \\
&= n \\
&= \mathsf{const}_n(k) \ .
\end{aligned}$$

Identity Function

- $\mathsf{id} : \mathbb{N} \to \mathbb{N}$, $\mathsf{id}(n) = n$ is primitive recursive:
  - $\mathsf{id} = \mathsf{proj}^1_0$:
    $\mathsf{proj}^1_0 : \mathbb{N}^1 \to \mathbb{N}$,
    $\mathsf{proj}^1_0(n) = n = \mathsf{id}(n)$.

Addition

- $\mathsf{add} : \mathbb{N}^2 \to \mathbb{N}$, $\mathsf{add}(x,y) = x + y$
  is primitive recursive.
  We have the laws:

$$\begin{aligned}
\mathsf{add}(x,0) &= x+0 \\
&= x \\
\mathsf{add}(x,y+1) &= x+(y+1) \\
&= (x+y)+1 \\
&= \mathsf{add}(x,y)+1
\end{aligned}$$

Addition

$$\begin{aligned} \mathsf{add}(x,0) &= x+0 \ , \\ \mathsf{add}(x,y+1) &= \mathsf{add}(x,y)+1 \ . \end{aligned}$$

- $\mathsf{add}(x,0) = g(x)$,
  where
  $g : \mathbb{N} \to \mathbb{N}$, $g(x) = x$,
  i.e. $g = \mathsf{id} = \mathsf{proj}^1_0$.

Addition

$$\begin{aligned} \mathsf{add}(x,0) &= x+0 = g(x) \ , \\ \mathsf{add}(x,y+1) &= \mathsf{add}(x,y)+1 \ . \end{aligned}$$

- $\mathsf{add}(x,y+1) = h(x,y,\mathsf{add}(x,y))$,
  where
  $h : \mathbb{N}^3 \to \mathbb{N}$, $h(x,y,z) := z+1$.
  $h = \mathsf{succ} \circ \mathsf{proj}^3_0$:

$$\begin{aligned} (\mathsf{succ} \circ \mathsf{proj}^3_2)(x,y,z) &= \mathsf{succ}(\mathsf{proj}^3_2(x,y,z)) \\ &= \mathsf{succ}(z) \\ &= z+1 \\ &= h(x,y,z) \ . \end{aligned}$$

Addition

$$\begin{aligned} \mathsf{add}(x,0) &= x+0 = g(x) \ , \\ \mathsf{add}(x,y+1) &= \mathsf{add}(x,y)+1 = h(x,y,\mathsf{add}(x,y)) \ , \\ g &= \mathsf{proj}^1_0 \ , \\ h &= \mathsf{succ} \circ \mathsf{proj}^3_2 \ . \end{aligned}$$

Therefore

$$\mathsf{add} = \mathsf{primrec}(\mathsf{proj}^1_0, \mathsf{succ} \circ \mathsf{proj}^3_2) \ .$$

Multiplication

- $\mathsf{mult} : \mathbb{N}^2 \to \mathbb{N}$, $\mathsf{mult}(x,y) = x \cdot y$
  is primitive recursive.
  We have the laws:

$$\begin{aligned} \mathsf{mult}(x,0) &= x \cdot 0 = 0 \\ \mathsf{mult}(x,y+1) &= x \cdot (y+1) \\ &= x \cdot y + x \\ &= \mathsf{mult}(x,y) + x \\ &= \mathsf{add}(\mathsf{mult}(x,y),x) \end{aligned}$$

Jump over rest

Multiplication

$$\begin{aligned} \mathsf{mult}(x,0) &= 0 \ , \\ \mathsf{mult}(x,y+1) &= \mathsf{add}(\mathsf{mult}(x,y),x) \ . \end{aligned}$$

- $\mathsf{mult}(x,0) = g(x)$, where $g : \mathbb{N} \to \mathbb{N}$, $g(x) = 0$, i.e. $g = \mathsf{zero}$,

Multiplication

$$\begin{aligned} \mathsf{mult}(x,0) &= 0 = g(x) \ , \\ \mathsf{mult}(x,y+1) &= \mathsf{add}(\mathsf{mult}(x,y),x) = h(x,y,\mathsf{mult}(x,y)) \ , \\ g &= \mathsf{zero} \ , \\ h &= \mathsf{add} \circ (\mathsf{proj}^3_2, \mathsf{proj}^3_0) \ . \end{aligned}$$

Therefore

$$\mathsf{mult} = \mathsf{primrec}(\mathsf{zero}, \mathsf{add} \circ (\mathsf{proj}^3_2, \mathsf{proj}^3_0)) \ .$$

Multiplication

$$\begin{aligned} \mathsf{mult}(x,0) &= 0 = g(x) \ , \\ \mathsf{mult}(x,y+1) &= \mathsf{add}(\mathsf{mult}(x,y),x) \ . \end{aligned}$$

- $\mathsf{mult}(x,y+1) = h(x,y,\mathsf{mult}(x,y))$, where $h : \mathbb{N}^3 \to \mathbb{N}$, $h(x,y,z) := \mathsf{add}(z,x)$. $h = \mathsf{add} \circ (\mathsf{proj}^3_2, \mathsf{proj}^3_0)$:

$$\begin{aligned} (\mathsf{add} \circ (\mathsf{proj}^3_2, \mathsf{proj}^3_0))(x,y,z) &= \mathsf{add}(\mathsf{proj}^3_2(x,y,z), \mathsf{proj}^3_0(x,y,z)) \\ &= \mathsf{add}(z,x) \\ &= h(x,y,z) \ . \end{aligned}$$

Predecessor Function

- pred is prim. rec.:

$$\begin{aligned} \mathsf{pred}(0) &= 0 \ , \\ \mathsf{pred}(x+1) &= x \ , \end{aligned}$$

Subtraction

- $\mathsf{sub}(x, y) = x \dot{-} y$ is prim. rec.:

$$\begin{aligned} \mathsf{sub}(x, 0) &= x \ , \\ \mathsf{sub}(x, y+1) &= \mathsf{pred}(\mathsf{sub}(x, y)) \ . \end{aligned}$$

Signum Function

- Note that

$$\mathsf{sig} = \chi_{x>0}$$

where $x > 0$ stands for the unary predicate, which is true for x iff $x > 0$:

$$\chi_{x>0}(y) = \left\{ \begin{array}{ll} 1, & \text{if } y > 0, \\ 0, & \text{if } y = 0. \end{array} \right\} = \mathsf{sig}(y)$$

Signum Function

- $\mathsf{sig} : \mathbb{N} \to \mathbb{N}$,

$$\mathsf{sig}(x) := \left\{ \begin{array}{ll} 1, & \text{if } x > 0, \\ 0, & \text{if } x = 0 \end{array} \right.$$

is prim. rec.:
$\mathsf{sig}(x) = x \dot{-} (x \dot{-} 1)$:

- For $x = 0$ we have

$$\begin{aligned} x \dot{-} (x \dot{-} 1) &= 0 \dot{-} (0 \dot{-} 1) = 0 \dot{-} 0 \\ &= 0 = \mathsf{sig}(x) \ . \end{aligned}$$

- For $x > 0$ we have

$$\begin{aligned} x \dot{-} (x \dot{-} 1) &= x - (x - 1) = x - x + 1 \\ &= 1 = \mathsf{sig}(x) \ . \end{aligned}$$

$x < y$ is Prim.-Rec.

$A(x, y) :\Leftrightarrow x < y$ is primitive recursive, since
$\chi_A(x, y) = \mathsf{sig}(y \dot{-} x)$:

- If $x < y$, then

$$y \dot{-} x = y - x > 0 \ ,$$

therefore

$$\mathsf{sig}(y \dot{-} x) = 1 = \chi_A(x, y)$$

- If $\neg(x < y)$, i.e. $x \geq y$, then

$$y \dot{-} x = 0 \ ,$$

$$\mathsf{sig}(y \dot{-} x) = 0 = \chi_A(x, y) \ .$$

Add., Mult., Exp.

- Consider the sequence of definitions of addition, multiplication, exponentiation:
  - **Addition:**

$$\begin{aligned} n+0 &= n \ , \\ n+(m+1) &= (n+m)+1 \ , \end{aligned}$$

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

$$n+m = ((+)\ 1)^m(n) \ .$$

Add., Mult., Exp.

- **Multiplication:**

$$\begin{aligned} n \cdot 0 &= 0 \ , \\ n \cdot (m+1) &= (n \cdot m) + n \ , \end{aligned}$$

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

$$n \cdot m = ((+)\ n)^m(0) \ .$$

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

$$f^n(m) := \underbrace{f(f(\cdots f}_{n \text{ times}}(m)\cdots)) \ )$$

  - This is not to be confused with exponentiation

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

- So

$$\begin{aligned} ((+)\ 1)^m(n) &= \underbrace{((+)\ 1)(((+)\ 1)(\cdots((+)\ 1)}_{m \text{ times}}(n)\cdots)) \\ &= (\cdots((m \underbrace{+1)+1)\cdots+1)}_{m \text{ times}} = m+n \end{aligned}$$

Add., Mult., Exp.

- **Exponentiation:**

$$\begin{aligned} n^0 &= 1 \ , \\ n^{m+1} &= (n^m) \cdot n \ , \end{aligned}$$

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:**

$$\begin{aligned} \mathsf{superexp}(n,0) &= 1 \ , \\ \mathsf{superexp}(n,m+1) &= n^{\mathsf{superexp}(n,m)} \ , \end{aligned}$$

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

$$n^m = ((\uparrow)\ n)^m(1) \ .$$

Supersuperexponentiation

- **Supersuperexponentiation:**

$$\begin{aligned} \mathsf{supersuperexp}(n,0) &= 1 \ , \\ \mathsf{supersuperexp}(n,m+1) &= \mathsf{superexp}(n, \mathsf{supersuperexp}(n,m)) \ , \end{aligned}$$

  - 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 Subsect. (c).

(b) Closure of the Prim.-Rec. Fun

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:
  - $(R \cup S)(\vec{x}) \Leftrightarrow R(\vec{x}) \vee S(\vec{x})$,
  - $(R \cap S)(\vec{x}) \Leftrightarrow R(\vec{x}) \wedge S(\vec{x})$,
  - $(\mathbb{N}^n \setminus R)(\vec{x}) \Leftrightarrow \neg R(\vec{x})$.
- So the prim.-rec. predicates are closed under the propositional connectives $\wedge$, $\vee$, $\neg$.
- **Example:**
  - Above we have seen that "$x < y$" is primitive recursive.
  - Therefore the predicates "$x \leq y$" and "$x = y$" are primitive recursive:
    - $x \leq y \Leftrightarrow \neg(y < x)$.
    - $x = y \Leftrightarrow x \leq y \wedge y \leq x$.

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

- Proof of $(R \cup S)(\vec{x}) \Leftrightarrow R(\vec{x}) \vee S(\vec{x})$:

$$\begin{aligned}(R \cup S)(\vec{x}) &\Leftrightarrow \vec{x} \in R \cup S \\ &\Leftrightarrow \vec{x} \in R \vee \vec{x} \in S \\ &\Leftrightarrow R(\vec{x}) \vee S(\vec{x})\end{aligned}$$

Jump over Rest

- Proof of $(R \cap S)(\vec{x}) \Leftrightarrow R(\vec{x}) \wedge S(\vec{x})$:

$$\begin{aligned}(R \cap S)(\vec{x}) &\Leftrightarrow \vec{x} \in R \cap S \\ &\Leftrightarrow \vec{x} \in R \wedge \vec{x} \in S \\ &\Leftrightarrow R(\vec{x}) \wedge S(\vec{x})\end{aligned}$$

Proof of Closure under $\cup$

- $\chi_{R\cup S}(\vec{x}) = \mathsf{sig}(\chi_R(\vec{x}) + \chi_S(\vec{x}))$,
  (therefore $R \cup S$ is primitive recursive):
  - If $R(\vec{x})$ holds, then

$$\underbrace{\mathsf{sig}(\underbrace{\underbrace{\chi_R(\vec{x})}_{=1} + \underbrace{\chi_S(\vec{x})}_{\geq 0}}_{\geq 1})}_{=1} = 1 = \chi_{R\cup S}(\vec{x}) \ .$$

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

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

$$\begin{aligned}(\mathbb{N}^n \setminus R)(\vec{x}) &\Leftrightarrow \vec{x} \in (\mathbb{N}^n \setminus R) \\ &\Leftrightarrow \vec{x} \notin R \\ &\Leftrightarrow \neg R(\vec{x})\end{aligned}$$

Proof of Closure under $\cup$

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

$$\underbrace{\mathsf{sig}(\underbrace{\underbrace{\chi_R(\vec{x})}_{\geq 0} + \underbrace{\chi_S(\vec{x})}_{=1}}_{\geq 1})}_{=1} = 1 = \chi_{R\cup S}(\vec{x})$$

Proof of Closure under $\cup$

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

$$\underbrace{\mathrm{sig}(\underbrace{\underbrace{\chi_R(\vec{x})}_{=0} + \underbrace{\chi_S(\vec{x})}_{=0}}_{=0})}_{=0} = 0 = \chi_{R\cup S}(\vec{x}) \ .$$

Proof of Closure under $\cap$

- If $\neg R(\vec{x})$ holds, then $\chi_R(\vec{x}) = 0$, therefore

$$\underbrace{\underbrace{\chi_R(\vec{x})}_{=0} \cdot \chi_S(\vec{x})}_{=0} = 0 = \chi_{R\cap S}(\vec{x}) \ .$$

- Similarly, if $\neg S(\vec{x})$, we have

$$\underbrace{\chi_R(\vec{x}) \cdot \underbrace{\chi_S(\vec{x})}_{=0}}_{=0} = 0 = \chi_{R\cap S}(\vec{x}) \ .$$

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

$$\underbrace{\underbrace{\chi_R(\vec{x})}_{=1} \cdot \underbrace{\chi_S(\vec{x})}_{=1}}_{=1} = 1 = \chi_{R\cap S}(\vec{x}) \ .$$

Proof of Closure under $\setminus$

- $\chi_{\mathbb{N}^n\setminus R}(\vec{x}) = 1 \dot{-} \chi_R(\vec{x})$
  (and therefore primitive recursive):
  Jump over Rest of Proof
  - If $R(\vec{x})$ holds, then $\chi_R(\vec{x}) = 1$, therefore

$$\underbrace{1 \dot{-} \underbrace{\chi_R(\vec{x})}_{=1}}_{=0} = 1 = \chi_{\mathbb{N}^n\setminus R}(\vec{x}) \ .$$

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

$$\underbrace{1 \dot{-} \underbrace{\chi_R(\vec{x})}_{=0}}_{=1} = 1 = \chi_{\mathbb{N}^n\setminus R}(\vec{x}) \ .$$

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

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

Show

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

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

$$\underbrace{\underbrace{g_1(\vec{x}) \cdot \underbrace{\chi_R(\vec{x})}_{=0}}_{=0} + \underbrace{g_2(\vec{x}) \cdot \underbrace{\chi_{\mathbb{N}^n \setminus R}(\vec{x})}_{=1}}_{=g_2(\vec{x})}}_{=g_2(\vec{x})} = g_2(\vec{x}) = f(\vec{x}) \ .$$

Definition by Cases

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

$f(\vec{x}) = g_1(\vec{x}) \cdot \chi_R(\vec{x}) + g_2(\vec{x}) \cdot \chi_{\mathbb{N}^n \setminus R}(\vec{x})$ prim. rec. :

Jump over rest of proof.

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

$$\underbrace{\underbrace{g_1(\vec{x}) \cdot \underbrace{\chi_R(\vec{x})}_{=1}}_{=g_1(\vec{x})} + \underbrace{g_2(\vec{x}) \cdot \underbrace{\chi_{\mathbb{N}^n \setminus R}(\vec{x})}_{=0}}_{=0}}_{=g_1(\vec{x})} = g_1(\vec{x}) = f(\vec{x}) \ .$$

Bounded Sums

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

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

  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

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

Proof that f is prim. rec.:

$$\begin{aligned} f(\vec{x}, 0) &= 0 \ , \\ f(\vec{x}, y+1) &= f(\vec{x}, y) + g(\vec{x}, y) \ . \end{aligned}$$

Jump over rest of proof.The last equations follows from

$$\begin{aligned} f(\vec{x}, y+1) &= \sum_{z<y+1} g(\vec{x}, z) \\ &= (\sum_{z<y} g(\vec{x}, z)) + g(\vec{x}, y) \\ &= f(\vec{x}, y) + g(\vec{x}, y) \ . \end{aligned}$$

Bounded Products

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

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

where

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

and for $y > 0$,

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

Omit Proof

Example

- We have above

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}, 0) \\ f(\vec{x}, 1) &= g(\vec{x}, 0) + g(\vec{x}, 1) \\ &= f(\vec{x}, 0) + g(\vec{x}, 0) \\ f(\vec{x}, 2) &= g(\vec{x}, 0) + g(\vec{x}, 1) + g(\vec{x}, 2) \\ &= f(\vec{x}, 1) + g(\vec{x}, 2) \end{aligned}$$

etc.

Bounded Products

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

Proof that f is prim. rec.:

$$\begin{aligned} f(\vec{x}, 0) &= 1 \ , \\ f(\vec{x}, y+1) &= f(\vec{x}, y) \cdot g(\vec{x}, y). \end{aligned}$$

Here, the last equations follows by

$$\begin{aligned} f(\vec{x}, y+1) &= \prod_{z<y+1} g(\vec{x}, z) \\ &= (\prod_{z<y} g(\vec{x}, z)) \cdot g(\vec{x}, y) \\ &= f(\vec{x}, y) \cdot g(\vec{x}, y) \ . \end{aligned}$$

Example

Example for closure under bounded products:
$f : \mathbb{N} \to \mathbb{N}$,

$$f(n) := n! = 1 \cdot 2 \cdot \cdots \cdot n$$

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

$$f(0) = 0! = 1 = \prod_{i<0}(i+1) \ .)$$

Bounded Quantification

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

$$R_1(\vec{x}, y) \ :\Leftrightarrow \ \forall z < y.R(\vec{x}, z) \ ,$$
$$R_2(\vec{x}, y) \ :\Leftrightarrow \ \exists z < y.R(\vec{x}, z) \ .$$

Remark on Factorial Function

- Alternatively, the factorial function can be defined directly by using primitive recursion as follows:

$$\begin{aligned} 0! &= 1 \\ (n+1)! &= n! \cdot (n+1) \end{aligned}$$

Bounded Quantification

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

Proof for R_1:

$$\chi_{R_1}(\vec{x}, y) = \prod_{z<y} \chi_R(\vec{x}, z) \quad :$$

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

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

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

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

Proof for R_2:

$$\chi_{R_2}(\vec{x}, y) = \text{sig}(\sum_{z<y} \chi_R(\vec{x}, z)) \quad :$$

Jump over Rest of Proof

- If $\forall z < y.\neg R(\vec{x}, z)$, then

$$\begin{aligned} \text{sig}(\sum_{z<y} \chi_R(\vec{x}, y)) &= \text{sig}(\sum_{z<y} 0) \\ &= \text{sig}(0) \\ &= 0 \\ &= \chi_{R_2}(\vec{x}, y) \ . \end{aligned}$$

Bounded Search

If $R \subseteq \mathbb{N}^{n+1}$ is a prim. rec. predicate, so is
$f(\vec{x}, y) := \mu z < y.R(\vec{x}, z)$, where

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

Bounded Search

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

Proof:

Define

$$\begin{aligned} Q(\vec{x}, y) &:\Leftrightarrow R(\vec{x}, y) \wedge \forall z < y.\neg R(\vec{x}, z) \ , \\ Q'(\vec{x}, y) &:\Leftrightarrow \forall z < y.\neg R(\vec{x}, z) \end{aligned}$$

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

$Q(\vec{x}, y) :\Leftrightarrow R(\vec{x}, y) \wedge \forall z < y.\neg R(\vec{x}, z)$,
$Q'(\vec{x}, y) :\Leftrightarrow \forall z < y.\neg R(\vec{x}, z)$,
Show $f(\vec{x}, y) = (\sum_{z<y} \chi_Q(\vec{x}, z) \cdot z) + \chi_{Q'}(\vec{x}, y) \cdot y$.

- Assume $\forall z < y.\neg R(\vec{x}, z)$.
  $\Rightarrow \neg Q(\vec{x}, z)$ for $z < y$,
  $\Rightarrow \forall z < y.\chi_Q(\vec{x}, z) \cdot z = 0$.
  Furthermore, $Q'(\vec{x}, y)$,
  therefore $\chi_{Q'}(\vec{x}, y) \cdot y = y$.
  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

$Q(\vec{x}, y) :\Leftrightarrow R(\vec{x}, y) \wedge \forall z < y.\neg R(\vec{x}, z)$,
$Q'(\vec{x}, y) :\Leftrightarrow \forall z < y.\neg R(\vec{x}, z)$,
Show $f(\vec{x}, y) = (\sum_{z<y} \chi_Q(\vec{x}, z) \cdot z) + \chi_{Q'}(\vec{x}, y) \cdot y$.

- Assume $\exists z < y.R(\vec{x}, z)$.
  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

  $$(\sum_{z<y} \chi_Q(\vec{x}, z) \cdot z) + \chi_{Q'}(\vec{x}, y) \cdot y = z = \mu z' < y.R(\vec{x}, z') \ .$$

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:

$$\begin{aligned} f(\vec{x}, 0) &= 0 \\ f(\vec{x}, y+1) &= \begin{cases} f(\vec{x}, y) & \text{if } f(\vec{x}, y) < y, \\ y & \text{if } f(\vec{x}, y) = y \wedge R(\vec{x}, y), \\ y+1 & \text{otherwise.} \end{cases} \end{aligned}$$

- 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

$$\begin{aligned} f &: \mathbb{N} \to \mathbb{N} \ , \\ f(x) &:= |\{y < x \mid P(y)\}| \ . \end{aligned}$$

- $f(x)$ 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 $P(y)$:

$$\begin{aligned} P(y) :\Leftrightarrow \\ & Q(y) \wedge (\exists z < y.Q(z)) \wedge \\ & \neg(\exists z < y.\exists z' < y.(Q(z) \wedge Q(z') \wedge z \neq z')) \end{aligned}$$

  $P(y)$ is primitive recursive, since it is defined from Q using $\wedge$, $\neg$, 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

$$P'(y) :\Leftrightarrow Q(y) \wedge \exists z < y.Q(z) \ .$$

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

Lemma 5.3

The following functions are primitive recursive:

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

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

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

Lemma 5.3

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

$$f(x,k,i) = \begin{cases} \pi_i^k(x), & \text{if } i < k, \\ x, & \text{otherwise.} \end{cases}$$

(Remember that $\pi_i^k(\pi^k(n_0,\ldots,n_{k-1})) = n_i$ for $i < k$.)
We write $\pi_i^k(a)$ 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) lh : $\mathbb{N} \to \mathbb{N}$.
(Remember that $\text{lh}(\langle x_0,\ldots,x_{k-1}\rangle) = k$.)

Proof of Lemma 5.3 (a), (b)

(a)

$$\begin{aligned} \pi(n,m) &= (\sum_{i \leq n+m} i) + m \\ &= (\sum_{i < n+m+1} i) + m \end{aligned}$$

is primitive-recursive.

(b) One can easily show that $n, m \leq \pi(n,m)$.
Therefore we can define

$$\begin{aligned} \pi_0(n) &:= \mu k < n+1.\exists l < n+1.n = \pi(k,l) \ , \\ \pi_1(n) &:= \mu l < n+1.\exists k < n+1.n = \pi(k,l) \ . \end{aligned}$$

Therefore π_0, π_1 are primitive-recursive.

Lemma 5.3

(g) $g : \mathbb{N}^2 \to \mathbb{N}$, $g(x,i) = (x)_i$.
(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 5.3 (c)

(c) Proof by induction on k:

- $k = 1$: $\pi^1(x) = x$, so π^1 is primitive-recursive.
- $k \to k+1$: Assume that π^k is primitive-recursive.
  Show that π^{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 π^{k+1} is primitive-recursive
(using that π, π^k are primitive-recursive).

Proof of Lemma 5.3 (d)

(d) We have

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

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 5.3 (d)

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

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

So f is primitive-recursive.

Proof of Lemma 5.3 (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 $g : \mathbb{N}^2 \to \mathbb{N}$,

$$\begin{aligned} g(x,0) &:= x \ , \\ g(x,k+1) &:= \pi_0(g(x,k)) \ , \end{aligned}$$

which is primitive-recursive.

Proof of Lemma 5.3 (e), (f), (g)

(e)

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

is primitive-recursive.

(f)

$$\mathsf{lh}(x) = \begin{cases} 0, & \text{if } x = 0, \\ \pi_0(x \dot{-} 1) + 1, & \text{if } x \neq 0. \end{cases}$$

(g)

$$\begin{aligned} (x)_i &= \pi_i^{\mathsf{lh}(x)}(\pi_1(x \dot{-} 1)) \\ &= f(\pi_1(x \dot{-} 1), \mathsf{lh}(x), i) \end{aligned}$$

is primitive-recursive.

Lemma and Definition 5.4

Prim. rec. functions as follows do exist:

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

$$\mathsf{snoc}(\langle x_0, \ldots, x_{n-1}\rangle, x) = \langle x_0, \ldots, x_{n-1}, x\rangle \ .$$

- **Remark:** snoc is the word cons reversed. snoc is like cons, but adds an element to the end rather than to the beginning of a list.

(b) last : $\mathbb{N} \to \mathbb{N}$ and beginning : $\mathbb{N} \to \mathbb{N}$ s.t.

$$\begin{aligned} \mathsf{last}(\mathsf{snoc}(x, y)) &= y \ , \\ \mathsf{beginning}(\mathsf{snoc}(x, y)) &= x \ . \end{aligned}$$

The proof will be omitted in the lecture.

Jump over proof.

Proof of Lemma 5.4 (a)

We have

$$\begin{aligned} &\mathsf{snoc}(\langle\rangle, y) \\ &= \mathsf{snoc}(0, y) \\ &= \langle y\rangle \ , \\ &\mathsf{snoc}(\langle x_0, \ldots, x_k\rangle, y) \\ &= \mathsf{snoc}(1 + \pi(k, \pi^{k+1}(x_0, \ldots, x_k)), y) \\ &= 1 + \pi(k+1, \pi(\pi_1((1 + \pi(k, \pi^{k+1}(x_0, \ldots, x_k))) \dot{-} 1), y)) \\ &\quad \text{(by } \mathsf{lh}(\langle x_0, \ldots, x_k\rangle) = k+1\text{)} \\ &= 1 + \pi(k+1, \pi(\pi_1(\pi(k, \pi^{k+1}(x_0, \ldots, x_k))), y)) \\ &= 1 + \pi(k+1, \pi(\pi^{k+1}(x_0, \ldots, x_k), y)) \\ &= 1 + \pi(k+1, \pi^{k+2}(x_0, \ldots, x_k, y)) \\ &= \langle x_0, \ldots, x_k, y\rangle \ . \end{aligned}$$

Proof of Lemma 5.4 (a)

Define

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

so snoc is primitive-recursive.

Proof of Lemma 5.4 (b)

Proof for beginning**:**
Define

$$\mathsf{beginning}(x) := \begin{cases} \langle\rangle, & \text{if } \mathsf{lh}(x) \leq 1, \\ \langle (x)_0\rangle & \text{if } \mathsf{lh}(x) = 2, \\ 1 + \pi((\mathsf{lh}(x) \dot{-} 1) \dot{-} 1, \pi_0(\pi_1(y \dot{-} 1))), & \text{otherwise.} \end{cases}$$

Proof of Lemma 5.4 (b)

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

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

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

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

$$\begin{aligned} \mathsf{beginning}(x) &= \langle\rangle \\ &= y \end{aligned}$$

Proof of Lemma 5.4 (b)

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

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

Then

$$\mathsf{snoc}(y, z) = 1 + \pi(n + 2, \pi(\pi_1(y \dot{-} 1), z)) \ .$$

Proof of Lemma 5.4 (b)

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

$$\begin{aligned} \mathsf{beginning}(x) &= \langle (x)_0 \rangle \\ &= \langle (\langle y', z \rangle)_0 \rangle \\ &= \langle y' \rangle \\ &= y \end{aligned}$$

Proof of Lemma 5.4 (b)

Therefore

$$\begin{aligned} &\mathsf{beginning}(\mathsf{snoc}(y, z)) \\ = \ & 1 + \pi(((\mathsf{lh}(x) \dot{-} 1) \dot{-} 1), \pi_0(\pi_1(\mathsf{snoc}(y, z) \dot{-} 1))) \\ = \ & 1 + \pi(n, \pi_0(\pi_1((1 + \pi(n + 2, \pi(\pi_1(y \dot{-} 1), z))) \dot{-} 1))) \\ = \ & 1 + \pi(n, \pi_0(\pi_1(\pi(n + 2, \pi(\pi_1(y \dot{-} 1), z))))) \\ = \ & 1 + \pi(n, \pi_0(\pi(\pi_1(y \dot{-} 1), z))) \\ = \ & 1 + \pi(n, \pi_1(y \dot{-} 1)) \\ = \ & 1 + \pi(n, \pi_1((1 + \pi(n + 1, \pi^{n+2}(y_0, \ldots, y_{n+1}))) \dot{-} 1)) \\ & 1 + \pi(n, \pi_1(\pi(n + 1, \pi^{n+2}(y_0, \ldots, y_{n+1})))) \\ = \ & 1 + \pi(n, \pi^{n+2}(y_0, \ldots, y_{n+1}))) \\ = \ & y \ . \end{aligned}$$

Proof of Lemma 5.4 (b)

Proof for last:
Define

$$\mathsf{last}(x) := (x)_{\mathsf{lh}(x) \dot{-} 1}$$

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

$$\begin{aligned} \mathsf{last}(\mathsf{snoc}(y, z)) &= \mathsf{last}(\langle y_0, \ldots, y_{n-1}, z \rangle) \\ &= (\langle y_0, \ldots, y_{n-1}, z \rangle)_{\mathsf{lh}(\langle y_0, \ldots, y_{n-1}, z \rangle) \dot{-} 1} \\ &= (\langle y_0, \ldots, y_{n-1}, z \rangle)_n \\ &= z \ . \end{aligned}$$

Course-of-Value Prim. Recursion

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

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

then f is prim. rec.

Course-of-Value Prim. Recursion

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

$$g : \mathbb{N}^{n+2} \to \mathbb{N}$$

is primitive recursive.
Then

$$f : \mathbb{N}^{n+1} \to \mathbb{N}$$

$$f(\vec{x}, k) = g(\vec{x}, k, \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, k-1) \rangle)$$

is prim. rec.

Example

Fibonacci numbers are prim. rec.
$\mathsf{fib} : \mathbb{N} \to \mathbb{N}$ given by:

$$\begin{aligned} \mathsf{fib}(0) &:= 1 \ , \\ \mathsf{fib}(1) &:= 1 \ , \\ \mathsf{fib}(n) &:= \mathsf{fib}(n-1) + \mathsf{fib}(n-2), \text{ if } n > 1, \end{aligned}$$

Definable by course-of-value primitive recursion:

- Let $\overline{\mathsf{fib}}(n) := \langle \mathsf{fib}(0), \ldots, \mathsf{fib}(n-1) \rangle$. Then

$$\mathsf{fib}(n) = \begin{cases} 1 & \text{if } n \leq 1, \\ (\overline{\mathsf{fib}}(n))_{n-2} + (\overline{\mathsf{fib}}(n))_{n-1} & \text{otherwise.} \end{cases}$$

Proof

Proof that prim. rec. functions are closed under course-of-value primitive recursion:
Let

$$f(\vec{x}, y) := g(\vec{x}, k, \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, k-1)\rangle)$$

Show f is prim. rec.
Define $h : \mathbb{N}^{n+1} \to \mathbb{N}$,

$$h(\vec{x}, y) := \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1)\rangle \ .$$

(Especially, $h(\vec{x}, 0) = \langle\rangle$.)

Proof

$$\begin{aligned} f(\vec{x}, y) &:= g(\vec{x}, k, \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, k-1)\rangle) \\ h(\vec{x}, y) &:= \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1)\rangle \ . \end{aligned}$$

h is prim. rec.

Now we have that

$$\begin{aligned} f(\vec{x}, y) &= (\langle f(\vec{x}, 0), \ldots, f(\vec{x}, y)\rangle)_y \\ &= (h(\vec{x}, y+1))_y \end{aligned}$$

is primitive recursive.

Proof

$$\begin{aligned} f(\vec{x}, y) &:= g(\vec{x}, k, \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, k-1)\rangle) \\ h(\vec{x}, y) &:= \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1)\rangle \ . \end{aligned}$$

$$\begin{aligned} h(\vec{x}, 0) &= \langle\rangle \ , \\ h(\vec{x}, y+1) &= \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1), f(\vec{x}, y)\rangle \\ &= \mathsf{snoc}(\underbrace{\langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1)\rangle}_{=h(\vec{x}, y)}, f(\vec{x}, y)) \\ &= \mathsf{snoc}(h(\vec{x}, y), g(\vec{x}, y, \underbrace{\langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1)\rangle}_{=h(\vec{x}, y)})) \\ &= \mathsf{snoc}(h(\vec{x}, y), g(\vec{x}, y, h(\vec{x}, y))) \ . \end{aligned}$$

Therefore h is primitive recursive.

Lemma and Definition 5.6

There exists prim. rec. functions as follows:

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

$$\begin{aligned} &\mathsf{append}(\langle n_0, \ldots, n_{k-1}\rangle, \langle m_0, \ldots, m_{l-1}\rangle) \\ &\qquad = \langle n_0, \ldots, n_{k-1}, m_0, \ldots, m_{l-1}\rangle \ . \end{aligned}$$

We write $n * m$ for $\mathsf{append}(n, m)$.

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

$$\mathsf{subst}(\langle x_0, \ldots, x_{n-1}\rangle, i, y) = \langle x_0, \ldots, x_{i-1}, y, x_{i+1}, x_{i+2}, \ldots, x$$

and if $i \geq n$, then

$$\mathsf{subst}(\langle x_0, \ldots, x_{n-1}\rangle, i, y) = \langle x_0, \ldots, x_{n-1}\rangle \ .$$

We write $x[i/y]$ for $\mathsf{subst}(x, i, y)$.

Lemma and Definition 5.6

(c) subseq : $\mathbb{N}^3 \to \mathbb{N}$ 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 $i \geq n$,

$$\text{subseq}(\langle x_0, \ldots, x_{n-1}\rangle, i, j) = \langle\rangle \ .$$

Proof of Lemma 5.6 (a)

We have

$$\begin{aligned}
&\text{append}(\langle x_0, \ldots, x_n\rangle, 0)\\
&= \text{append}(\langle x_0, \ldots, x_n\rangle, \langle\rangle)\\
&= \langle x_0, \ldots, x_n\rangle \ ,\\
&\quad \text{and for } m > 0\\
&\text{append}(\langle x_0, \ldots, x_n\rangle, \langle y_0, \ldots, y_m\rangle)\\
&= \langle x_0, \ldots, x_n, y_0, \ldots, y_m\rangle\\
&= \text{snoc}(\langle x_0, \ldots, x_n, y_0, \ldots, y_{m-1}\rangle, y_m)\\
&= \text{snoc}(\text{append}(\langle x_0, \ldots, x_n\rangle, \langle y_0, \ldots, y_{m-1}\rangle), y_m)\\
&= \text{snoc}(\text{append}(\langle x_0, \ldots, x_n\rangle,\\
&\qquad \text{beginning}(\langle y_0, \ldots, y_m\rangle)),\\
&\qquad \text{last}(\langle y_0, \ldots, y_m\rangle)) \ .
\end{aligned}$$

Lemma and Definition 5.6

(d) half : $\mathbb{N} \to \mathbb{N}$,
s.t. $\text{half}(n) = k$ if $n = 2k$ or $n = 2k + 1$.

(e) The function bin : $\mathbb{N} \to \mathbb{N}$, 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 $\text{bin}^{-1} : \mathbb{N} \to \mathbb{N}$, 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 5.6 (a)

Therefore we have

$$\begin{aligned}
\text{append}(x, 0) &= x \ ,\\
\text{append}(x, y) &= \text{snoc}(\text{append}(x, \text{beginning}(y)), \text{last}(y)) \ ,
\end{aligned}$$

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

Proof of Lemma 5.6 (b)

We have

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

Therefore subst is definable by course-of-value primitive recursion.

Proof of Lemma 5.6 (d), (e)

(d) $\mathsf{half}(x) = \mu y < x.(2 \cdot y = x \vee 2 \cdot y + 1 = x)$.

(e)

$$\mathsf{bin}(x) = \begin{cases} \langle 0 \rangle, & \text{if } x = 0, \\ \langle 1 \rangle & \text{if } x = 1, \\ \mathsf{snoc}(\mathsf{half}(x), x \dot{-} (2 \cdot \mathsf{half}(x))), & \text{if } x > 1. \end{cases}$$

therefore definable by course-of-value primitive recursion.

Proof of Lemma 5.6 (c)

We can define

$$\mathsf{subseq}(x,i,j) = \begin{cases} \langle\rangle, & \text{if } i \ge \mathsf{lh}(x), \\ \mathsf{subseq}(\mathsf{beginning}(x), i, j), & \text{if } i < \mathsf{lh}(x) \\ & \text{and } j < \mathsf{lh}(x), \\ \mathsf{snoc}(\mathsf{subseq}(\mathsf{beginning}(x), i, j), \mathsf{last}(x)) & \text{if } i < \mathsf{lh}(x) \le j, \end{cases}$$

which is a definition by course-of-value primitive recursion.

Proof of Lemma 5.6 (f)

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

therefore definable by course-of-value primitive recursion.

(c) The Part.-Rec. Functions

Ackermann Function

- At the end of Subsect. (b) we have started to introduce a series of functions

  $\text{add} \longrightarrow \text{mult} \longrightarrow \exp \longrightarrow \text{superexp} \longrightarrow \text{supersuperexp} \longrightarrow \cdots$

  where each function in this sequence is defined by primitive recursion using the previous function.
- Such a sequence will eventually exhaust the primitive recursive function.

The Ackermann Function

- Traditionally, instead of defining a sequence of binary functions one defines a sequence of unary functions with similar growth rate, the **Ackermann function**.
- The Ackermann function exhausts all primitive-recursive functions.
- The uniform version of the Ackermann function will therefore no longer be primitive recursive.
- In order to obtain a complete model of computation, we will need to extend the primitive-recursive function by closure under μ.
  - The resulting functions will be called the **partial recursive functions**.

The Ackermann Function

- Let $n \in \mathbb{N}$.
  The **n-th branch of the Ackermann function**
  $\text{Ack}_n : \mathbb{N} \longrightarrow \mathbb{N}$, is defined by

$$\begin{aligned} \text{Ack}_0(y) &= y+1 \ , \\ \text{Ack}_{n+1}(y) &= (\text{Ack}_n)^{y+1}(1) := \underbrace{\text{Ack}_n(\text{Ack}_n(\cdots \text{Ack}_n(}_{y+1 \text{ times}}1))) \ . \end{aligned}$$

Ack_n is prim. rec. **for fixed** n.

Examples

$$\begin{aligned} \text{Ack}_0(n) &= n+1 \ . \\ \text{Ack}_1(n) &= \text{Ack}_0^{n+1}(1) \\ &= 1\underbrace{+1+\cdots+1}_{n+1 \text{ times}} \\ &= 1+n+1 = n+2 \ . \\ \text{Ack}_2(n) &= \text{Ack}_1^{n+1}(1) \\ &= 1\underbrace{+2+\cdots+2}_{n+1 \text{ times}} \\ &= 1+2(n+1) \\ &= 2n+3 > 2n \ . \end{aligned}$$

Examples

$$\mathsf{Ack}_2(n) > 2n \ .$$

$$\begin{aligned}
\mathsf{Ack}_3(n) &= \mathsf{Ack}_2^{n+1}(1) \\
&> \underbrace{2 \cdot 2 \cdot \cdots \cdot 2}_{n+1 \text{ times}} \cdot 1 \\
&= 2^{n+1} > 2^n \ . \\
\mathsf{Ack}_4(n) &= \mathsf{Ack}_3^{n+1}(1) \\
&> \underbrace{2^{\cdots^{2^1}}}_{n+1 \text{ times}} \ .
\end{aligned}$$

The Ackerm. Funct. is Prim.-Rec.

$$\begin{aligned}
\mathsf{Ack}_0(y) &= y+1 \ , \\
\mathsf{Ack}_{n+1}(y) &= (\mathsf{Ack}_n)^{y+1}(1) \ .
\end{aligned}$$

Proof that Ack_n is prim. rec. by Induction(n):

- **Base-case:**
  $\mathsf{Ack}_0 = \mathsf{succ}$ is prim. rec.
- **Induction step:**
  Assume Ack_n is primitive recursive.
  Show Ack_{n+1} is primitive recursive.
  We have:

$$\begin{aligned}
\mathsf{Ack}_{n+1}(0) &= \mathsf{Ack}_n(1) \ , \\
\mathsf{Ack}_{n+1}(y+1) &= \mathsf{Ack}_n(\mathsf{Ack}_{n+1}(y)) \ .
\end{aligned}$$

Examples

$$\mathsf{Ack}_4(n) > \underbrace{2^{\cdots^{2^1}}}_{n+1 \text{ times}} \ .$$

- $\mathsf{Ack}_5(n)$ will iterate Ack_4 $n+1$ times, etc.
- So even for very small n, $\mathsf{Ack}_5(n)$ will exceed the number of particles in the universe, and Ack_5 is therefore not realistically computable.

The Ackermann Function

- Therefore Ack_{n+1} is primitive recursive.
  End of proof.
- **Remark:**
  - Ack_n for **fixed** n is **primitive recursive**.
  - However a **uniform** version of the Ackermann function is **not primitive recursive**.

The Uniform Ackermann Function

The uniform version of the **Ackermann function**
$\mathsf{Ack} : \mathbb{N}^2 \to \mathbb{N}$ is defined as

$$\mathsf{Ack}(n, m) := \mathsf{Ack}_n(m) \ .$$

Therefore we have the equations

$$\begin{aligned} \mathsf{Ack}(0, y) &= y + 1 \ , \\ \mathsf{Ack}(x + 1, 0) &= \mathsf{Ack}(x, 1) \ , \\ \mathsf{Ack}(x + 1, y + 1) &= \mathsf{Ack}(x, \mathsf{Ack}(x + 1, y)) \ . \end{aligned}$$

In order to show that Ack is not prim.-rec., we show first the following technical lemma:

Proof of Lemma 5.8 (a)

Induction on m.
$m = 0$:

$$\mathsf{Ack}(0, n) = n + 1 > n \ .$$

$m \to m + 1$: Side-induction on n
$n = 0$:

$$\mathsf{Ack}(m + 1, 0) = \mathsf{Ack}(m, 1) \overset{\text{IH}}{>} 1 > 0 \ .$$

Lemma 5.8

For each n, m, the following holds:

(a) $\mathsf{Ack}(m, n) > n$.

(b) Ack_m is strictly monotone,
i.e. $\mathsf{Ack}(m, n + 1) > \mathsf{Ack}(m, n)$.

(c) $\mathsf{Ack}(m + 1, n) > \mathsf{Ack}(m, n)$.

(d) $\mathsf{Ack}(m, \mathsf{Ack}(m, n)) < \mathsf{Ack}(m + 2, n)$.

(e) $\mathsf{Ack}(m, 2n) < \mathsf{Ack}(m + 2, n)$.

(f) $\mathsf{Ack}(m, 2^k \cdot n) < \mathsf{Ack}(m + 2k, n)$.

The proof will be omitted in the lecture.

Jump over proof.

Proof of Lemma 5.8 (a)

$n \to n + 1$:

$$\begin{aligned} \mathsf{Ack}(m + 1, n + 1) &= \mathsf{Ack}(m, \mathsf{Ack}(m + 1, n)) \\ &\overset{\text{Main IH}}{>} \mathsf{Ack}(m + 1, n) \\ &\overset{\text{Side IH}}{>} n \ , \end{aligned}$$

therefore

$$\mathsf{Ack}(m + 1, n + 1) > n + 1 \ .$$

Proof of Lemma 5.8 (b)

Case $m = 0$:

$$\mathsf{Ack}(0, n+1) = n+2 > n+1 = \mathsf{Ack}(0, n) \ .$$

Case $m = m' + 1$:

$$\begin{aligned} \mathsf{Ack}(m'+1, n+1) &= \mathsf{Ack}(m', \mathsf{Ack}(m'+1, n)) \\ &\overset{\text{(a)}}{>} \mathsf{Ack}(m'+1, n) \ . \end{aligned}$$

Proof of Lemma 5.8 (c)

$m \to m+1$: Side-induction on n:
$n = 0$:

$$\mathsf{Ack}(m+2, 0) = \mathsf{Ack}(m+1, 1) \overset{\text{(b)}}{>} \mathsf{Ack}(m+1, 0) \ .$$

$n \to n+1$:

$$\begin{aligned} &\mathsf{Ack}(m+2, n+1) \\ &\quad = \mathsf{Ack}(m+1, \mathsf{Ack}(m+2, n)) \\ &\quad \overset{\text{main-IH}}{>} \mathsf{Ack}(m, \mathsf{Ack}(m+2, n)) \\ &\quad \overset{\text{side-IH + (b)}}{>} \mathsf{Ack}(m, \mathsf{Ack}(m+1, n)) = \mathsf{Ack}(m+1, n+1) \ . \end{aligned}$$

Proof of Lemma 5.8 (c)

Induction on m.
$m = 0$:

$$\mathsf{Ack}(1, n) = n+2 > n+1 = \mathsf{Ack}(0, n)$$

Proof of Lemma 5.8 (d)

Case $m = 0$:

$$\mathsf{Ack}(0, \mathsf{Ack}(0, n)) = n+2 < 2n+3 = \mathsf{Ack}(2, n) \ .$$

Assume now $m > 0$.
Proof of the assertion by induction on n:
$n = 0$:

$$\begin{aligned} \mathsf{Ack}(m+2, 0) &= \mathsf{Ack}(m+1, 1) \\ &= \mathsf{Ack}(m, (\mathsf{Ack}(m+1, 0)) \\ &> \mathsf{Ack}(m, \mathsf{Ack}(m, 0)) \ . \end{aligned}$$

Proof of Lemma 5.8 (d)

$n \to n+1$:

$$\begin{aligned}
\mathsf{Ack}(m+2,n+1) &= \mathsf{Ack}(m+1,\mathsf{Ack}(m+2,n)) \\
&\overset{\text{IH,(b)}}{>} \mathsf{Ack}(m+1,\mathsf{Ack}(m,\mathsf{Ack}(m,n))) \\
&\overset{\text{(b),(c)}}{>} \mathsf{Ack}(m,\mathsf{Ack}(m-1,\mathsf{Ack}(m,n))) \\
&= \mathsf{Ack}(m,\mathsf{Ack}(m,n+1)) \ .
\end{aligned}$$

Proof of Lemma 5.8 (e)

Case $m = m' + 1$:
Induction on n:
$n = 0$:

$\mathsf{Ack}(m'+1, 2n) = \mathsf{Ack}(m'+1, 0) < \mathsf{Ack}(m'+3, 0) = \mathsf{Ack}(m'+3, n)$

$n \to n+1$:

$$\begin{aligned}
\mathsf{Ack}(m'+1,2n+2) &= \mathsf{Ack}(m',\mathsf{Ack}(m',\mathsf{Ack}(m'+1,2n)) \\
&\overset{\text{(d)}}{<} \mathsf{Ack}(m'+2,\mathsf{Ack}(m'+1,2n)) \\
&\overset{\text{IH}}{<} \mathsf{Ack}(m'+2,\mathsf{Ack}(m'+3,n)) \\
&= \mathsf{Ack}(m'+3,n+1)
\end{aligned}$$

Proof of Lemma 5.8 (e)

Case $m = 0$:

$$\begin{aligned}
\mathsf{Ack}(m,2n) &= \mathsf{Ack}(0,2n) \\
&= 2n+1 \\
&< 2n+3 \\
&= \mathsf{Ack}(2,n) = \mathsf{Ack}(m+2,n) \ .
\end{aligned}$$

Proof of Lemma 5.8 (f)

Induction on k:
$k = 0$: trivial.
$k \to k+1$:

$$\begin{aligned}
\mathsf{Ack}(m,2^{k+1}\cdot n) &= \mathsf{Ack}(m,2\cdot 2^k\cdot n) \\
&\overset{\text{(e)}}{<} \mathsf{Ack}(m+2,2^k\cdot n) \\
&\overset{\text{IH}}{<} \mathsf{Ack}(m+2+2k,n) \\
&= \mathsf{Ack}(m+2(k+1),n) \ .
\end{aligned}$$

Lemma 5.9

Every primitive recursive function $f : \mathbb{N}^n \to \mathbb{N}$ can be majorised by one branch of the Ackermann function:

There exists N s.t.

$$f(x_0, \ldots, x_{n-1}) < \mathsf{Ack}_N(x_0 + \cdots + x_{n-1})$$

for all $x_0, \ldots, x_{n-1} \in \mathbb{N}$.

Especially, if $f : \mathbb{N} \to \mathbb{N}$ is prim. rec., then there exists an N s.t.

$$\forall x \in \mathbb{N}. f(x) < \mathsf{Ack}_N(x)$$

The proof will be omitted in the lecture.

Jump over proof.

Proof of Lemma 5.9

Basic functions:

zero:

$$\mathsf{zero}(x) = 0 < x + 1 = \mathsf{Ack}_0(x) \ .$$

succ:

$$\mathsf{succ}(x) = \mathsf{Ack}(0, x) < \mathsf{Ack}(1, x) = \mathsf{Ack}_1(x) \ .$$

proj_i^n:

$$\begin{aligned} \mathsf{proj}(x_0, \ldots, x_{n-1}) &= x_i \\ &< x_0 + \cdots + x_{n-1} + 1 \\ &= \mathsf{Ack}_0(x_0 + \cdots + x_{n-1}) \ . \end{aligned}$$

Proof of Lemma 5.9

We write, if $\vec{x} = x_0, \ldots, x_{n-1}$

$$\sum(\vec{x}) := x_0 + \cdots x_{n-1} \ .$$

We proof the assertion by induction on the definition of primitive-recursive functions.

Proof of Lemma 5.9

Composition: Assume assertion holds for $f : \mathbb{N}^k \to \mathbb{N}$ and $g_i : \mathbb{N}^n \to \mathbb{N}$. Show assertion for $h := f \circ (g_0, \ldots, g_{k-1})$. Assume

$$f(\vec{y}) < \mathsf{Ack}_l(\sum(\vec{y})) \ ,$$

and

$$g_i(\vec{x}) < \mathsf{Ack}_{m_i}(\sum(\vec{x})) \ .$$

Let $N := \max\{l, m_0, \ldots, m_{k-1}\}$. By Lemma 5.8 (c) it follows

$$f(\vec{y}) < \mathsf{Ack}_N(\sum(\vec{y})) \ ,$$

$$g_i(\vec{x}) < \mathsf{Ack}_N(\sum(\vec{x})) \ .$$

Proof of Lemma 5.9

Then, with M s.t. $l < 2^M$, we have

$$\begin{aligned}
h(\vec{x}) &= f(g_0(\vec{x}), \ldots, g_{k-1}(\vec{x})) \\
&< \mathsf{Ack}_N(g_0(\vec{x}) + \cdots + g_{k-1}(\vec{x})) \\
&\overset{\text{(b)}}{<} \mathsf{Ack}_N(\mathsf{Ack}_N(\textstyle\sum(\vec{x})) + \cdots + \mathsf{Ack}_N(\textstyle\sum(\vec{x}))) \\
&= \mathsf{Ack}_N(\mathsf{Ack}_N(\textstyle\sum(\vec{x})) \cdot k) \\
&< \mathsf{Ack}_N(\mathsf{Ack}_N(\textstyle\sum(\vec{x})) \cdot 2^M) \\
&< \mathsf{Ack}_{N+2M}(\mathsf{Ack}_N(\textstyle\sum(\vec{x}))) \\
&\leq \mathsf{Ack}_{N+2M}(\mathsf{Ack}_{N+2M}(\textstyle\sum(\vec{x}))) \\
&< \mathsf{Ack}_{N+2M+2}(\textstyle\sum(\vec{x})) \ .
\end{aligned}$$

Proof of Lemma 5.9

We show

$$h(\vec{x}, y) < \mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + y)$$

by induction on y:
$y = 0$:

$$\begin{aligned}
h(\vec{x}, 0) &= g(\vec{x}) \\
&< \mathsf{Ack}_N(\textstyle\sum(\vec{x})) \\
&< \mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + 0) \ .
\end{aligned}$$

Proof of Lemma 5.9

Primitive recursion, $n > 1$:
Assume assertion holds for $f : \mathbb{N}^n \to \mathbb{N}$ and $g : \mathbb{N}^{n+2} \to \mathbb{N}$.
Show assertion for $h := \mathsf{primrec}(f, g) : \mathbb{N}^{n+1} \to \mathbb{N}$.
Assume

$$f(\vec{x}) < \mathsf{Ack}_l(\textstyle\sum(\vec{x})) \ ,$$
$$g(\vec{x}, y, z) < \mathsf{Ack}_r(\textstyle\sum(\vec{x}) + y + z) \ .$$

Let $N := \max\{l, r\}$, $k < 2^M$. Then

$$f(\vec{x}) < \mathsf{Ack}_N(\textstyle\sum(\vec{x})) \ ,$$
$$g(\vec{x}, y, z) < \mathsf{Ack}_N(\textstyle\sum(\vec{x}) + y + z) \ .$$

Proof of Lemma 5.9

$y \to y + 1$:

$$\begin{aligned}
h(\vec{x}, y+1) &= g(\vec{x}, y, h(\vec{x}, y)) \\
&< \mathsf{Ack}_N(\textstyle\sum(\vec{x}) + y + h(\vec{x}, y)) \\
&\overset{\text{IH}}{<} \mathsf{Ack}_N(\textstyle\sum(\vec{x}) + y + \mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + y)) \\
&< \mathsf{Ack}_N(\mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + y) + \mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + \\
&= \mathsf{Ack}_N(2 \cdot \mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + y)) \\
&< \mathsf{Ack}_{N+2}(\mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + y)) \\
&= \mathsf{Ack}_{N+3}(\textstyle\sum(\vec{x}) + y + 1)
\end{aligned}$$

Proof of Lemma 5.9

Primitive recursion $n = 0$:
Assume $l \in \mathbb{N}$, $g : \mathbb{N}^2 \to \mathbb{N}$. Show assertion for
$h := \mathsf{primrec}(l, g) : \mathbb{N}^1 \to \mathbb{N}$.
Define

$$\begin{array}{lcll} f' & : & \mathbb{N} \to \mathbb{N}, & f'(x) = l \ , \\ g' & : & \mathbb{N}^3 \to \mathbb{N}, & g'(x,y,z) = g(y,z) \ , \\ h' & : & \mathbb{N}^2 \to \mathbb{N}, & h := \mathsf{primrec}(f', g') \ . \end{array}$$

Using the constructions already shown and IH it follows that

$$h'(x,y) < \mathsf{Ack}_N(x+y)$$

for some N. Therefore

$$h(y) = h'(0,y) < \mathsf{Ack}_N(y) \ .$$

Remark

Direct argument for the existence of non-primitive recursive computable functions:
Assume all computable functions are prim. rec..
Define $h : \mathbb{N}^2 \to \mathbb{N}$,

$$h(e,n) = \begin{cases} f(n), & \text{if } e \text{ encodes a string in ASCII} \\ & \text{which is a term denoting a unary} \\ & \text{primitive recursive function } f, \\ 0, & \text{otherwise.} \end{cases}$$

h is computable, therefore primitive recursive.

h can be considered as an **interpreter** for the language of primitive recursive functions.

Lemma 5.10

The Ackermann function is **not primitive recursive**.

Proof:
Assume Ack were primitive recursive.
Then

$$f : \mathbb{N} \to \mathbb{N}, \qquad f(n) := \mathsf{Ack}(n,n)$$

is prim. rec.
Then there exists an $N \in \mathbb{N}$, s.t. $f(n) < \mathsf{Ack}(N,n)$ for all n.
Especially

$$\mathsf{Ack}(N,N) = f(N) < \mathsf{Ack}(N,N) \ ,$$

a contradiction.

Remark

If e code for f, then $h(e,n) = f(n)$.
h is prim. rec.
For every primitive recursive function f there exists a code e of f, therefore

$$\forall n. h(e,n) = f(n) \ ,$$

$$f = \lambda n. h(e,n) \ .$$

Define

$$f : \mathbb{N} \to \mathbb{N}, \qquad f(n) := h(n,n) + 1 \ .$$

f is defined in such a way, that it cannot be of the form $\lambda n. h(e,n)$: If it were, we would get

$$h(e,e) + 1 = f(e) = (\lambda n. h(e,n))(e) = h(e,e) \ .$$

Remark

If e code for f, then $h(e,n) = f(n)$.
h is prim. rec.
$f(n) := h(n,n) + 1$.
$f \neq \lambda n.h(e,n)$.
h is primitive recursive, therefore as well f.
Therefore $h = \lambda n.h(e,n)$, which cannot be the case.
Therefore we obtain a contradiction.

Extension of the above

The above proof shows as well that by adding the Ackermann function as basic function to the definition of primitive recursive functions, or any other functions, we still won't obtain all computable functions

– the above argument can be used for such sets of functions in just the same way as for the set of primitive recursive functions.

Extension of the above

The above proof can be used in other contexts as well.
It shows as well that there is no programming language, which computes all computable functions and such that all functions definable in this language are total.
If we had such a language, then we could define codes e for programs in this language,
and therefore we could define a computable $h : \mathbb{N}^2 \to \mathbb{N}$, s.t.

$$h(e,n) = \begin{cases} f(n), & \text{if } e \text{ is a program in this language} \\ & \text{for a unary function } f, \\ 0, & \text{otherwise.} \end{cases}$$

Using essentially the same argument as before we get a contradiction.

Limitations of Prim.-Rec. Functs.

We can define prim. rec. functions which

- compute the result of running n steps of a URM or TM,
- check whether a URM or TM has stopped,
- obtain, depending on n arguments the initial configuration for computing $\mathrm{U}^{(n)}$ for a URM U or $\mathrm{T}^{(n)}$ for a TM T,
- extract from a configuration of U and T, in which this machine has stopped, the result obtained by $\mathrm{U}^{(n)}$, $\mathrm{T}^{(n)}$.

Formally done for TMs in Subsection 6 (a).

Limitations of Prim.-Rec. Functs.

- However, TMs and URMs don't allow to compute an n, s.t. after n steps the URM or TM has stopped.
- Extension of prim. rec. functions by adding to the principles of forming functions closure under μ.
- Resulting set called the set of **partial recursive functions.**
- Using the μ-operator, we will be able to find the least n s.t. a TM or URM stops (and return $\perp$, if no such n exists.)
  - Therefore one can show that all TM- and URM-computable functions are partial recursive.

Partial Rec. Functions

Inductive definition of the set of **partial recursive functions** f together with their **arity**,

i.e. together with the k s.t. $f : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$.

We write "$f : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$ is partial recursive" for "f is partial recursive with arity k", and $\mathbb{N}$ for $\mathbb{N}^1$.

- The following **basic functions** are partial recursive:
  - zero : $\mathbb{N} \xrightarrow{\sim} \mathbb{N}$,
  - succ : $\mathbb{N} \xrightarrow{\sim} \mathbb{N}$,
  - $\text{proj}_i^k : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$ $(0 \leq i \leq k)$.

Limitations of Prim.-Rec. Functs.

- $\mu(f)$ might be partial, even if f is total.
  $\Rightarrow$ Resulting set is set of **partial** functions, therefore name **partial** recursive functions.
- The **recursive** functions will be the partial recursive functions, which are total.

Partial Rec. Functions

- If
  - $g : \mathbb{N}^k \xrightarrow{\sim} \mathbb{N}$ is partial recursive,
  - for $i = 0, \ldots, k-1$ we have $h_i : \mathbb{N}^n \xrightarrow{\sim} \mathbb{N}$ is partial recursive,

  then $g \circ (h_0, \ldots, h_{k-1}) : \mathbb{N}^n \xrightarrow{\sim} \mathbb{N}$ is partial recursive as well.

Partial Rec. Functions

- If
  - $g : \mathbb{N}^n \xrightarrow{\sim} \mathbb{N}$,
  - $h : \mathbb{N}^{n+2} \xrightarrow{\sim} \mathbb{N}$ are partial recursive,

  then $\mathsf{primrec}(g, h) : \mathbb{N}^{n+1} \xrightarrow{\sim} \mathbb{N}$ is partial recursive as well.
- If
  - $k \in \mathbb{N}$,
  - $h : \mathbb{N}^2 \xrightarrow{\sim} \mathbb{N}$ is partial recursive,

  then $\mathsf{primrec}(k, h) : \mathbb{N} \xrightarrow{\sim} \mathbb{N}$ is partial recursive as well.

Recursive Functions

(a) A **recursive function** is a partial recursive function, which is total.

(b) A **recursive relation** is a relation $R \subseteq \mathbb{N}^n$ s.t. χ_R is recursive.

Example:
Ack is recursive, but not primitive recursive.

Partial Rec. Functions

- If $g : \mathbb{N}^{k+2} \xrightarrow{\sim} \mathbb{N}$ is partial recursive,
  then $\mu(g) : \mathbb{N}^{k+1} \xrightarrow{\sim} \mathbb{N}$ is partial recursive as well.
  (Remember that $f := \mu(g)$ has defining equation

$$f(\vec{x}) \simeq \begin{cases} \min\{k \in \mathbb{N} \mid & \\ \qquad g(\vec{x}, k) \simeq 0 & \\ \qquad \wedge \forall l < k . g(\vec{x}, l) \downarrow\}, & \text{if such a } k \text{ exists,} \\ \bot, & \text{otherwise.)} \end{cases}$$

Closure of Part. Rec. Func.

- Every prim. rec. function (relation) is recursive.
- The recursive functions and relations have the same closure properties as those discussed for the prim. rec. functions and relations.

Closure of Part. Rec. Func.

Let for a predicate $U \subseteq \mathbb{N}^{n+1}$

$$\mu z.U(\vec{n},z) := \begin{cases} \min\{z \mid U(\vec{n},z)\}, & \text{if such a } z \text{ exists,} \\ \bot, & \text{otherwise.} \end{cases}$$

Then we have that, if U is recursive, so is the function

$$f(\vec{n}) :\simeq \mu z.U(\vec{n},z) \ .$$

Proof:

$$f(\vec{n}) \simeq \mu z.(\chi_{\mathbb{N}^n \setminus U}(\vec{n},z) \simeq 0) \ .$$