5. The Primitive Recursive Functior

- In this module we consider **3 models of computation**.
  - The **URMs**, which captures computation as it happens on a computer.
  - The **Turing Machines**, which capture computation on a piece of paper.
  - The **partial recursive functions**, developed in this and the next section.
    - Partial recursive functions were first proposed by Gödel and Kleene 1936.
- There are many other models of computation.

Algebraic View of Computation

- Main **motivation** for partial recursive functions:
  - **Algebraic view of computation.**
  - The class of partial computable functions in this model is defined by certain **combinators**.
    - We have some initial functions and close them under operations which form from partial computable functions new partial computable functions.
  - So in this model of computation we define directly a set of functions (rather than defining first a programming language and then the functions defined by it).

Algebraic View of Computation

- We can assign a term to each partial recursive function.
  - E.g.

$$\mathrm{primrec}(\mathrm{zero}, \mathrm{proj}^0_1)$$

    denotes the predecessor function.
- These combinators allow
  - to **define functions more easily** directly, and therefore show that they are computable;
  - and to **manipulate terms** denoting partial recursive functions.

Primitive Recursive Functions

- In this section we will first start introducing the **primitive recursive functions.**
- They form an important **subclass of the partial recursive functions.**
- Main property of the primitive recursiv functions.
  - All primitive recursive functions are **total**.
  - Therefore **not all computable functions** are **primitive recursive**.
    - There exists no programming language, such that all definable functions are total, which allows to define all computable functions.

Primitive Recursive Functions

- The primitive recursive functions contain all **feasible functions** (and many infeasible functions as well.
- Therefore all **realistic functions can be defined primitive recursively**.
- The principle of primitive recursion is closely related to the principle of **induction**.
  - In the dependently typed programming language Agda induction and primitive recursion are the same principle.
- Extensions of the principle of primitive recursion form the main ingredient of many **functional programming languages**.

Overview

(a) Introduction of **primitive recursive functions**.

(b) Closure Properties of the **primitive rec. functions**

- We will show that the set of primitive recursive functions is a rich set of functions, closed under many operations.
- This will show as well extend our intuition of how powerful URM computable functions are.

(a) Introd. of the Prim. Rec. Functio

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

  Remember that these functions have defining equations
  - $\mathrm{zero}(y) = 0$,
  - $\mathrm{succ}(y) = y + 1$,
  - $\mathrm{proj}_i^k(y_0, \ldots, y_{k-1}) = y_i$.

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
  - $g : \mathbb{N}^n \to \mathbb{N}$,
  - $h : \mathbb{N}^{n+2} \to \mathbb{N}$ 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 $\mathsf{primrec}(g, h)$.

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
  - $f(0) = k$,
  - $f(y + 1) = h(y, f(y))$.
- f is denoted by $\mathsf{primrec}(k, h)$.

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

Inductively Defined Sets

E.g.

- $$\underbrace{\text{primrec}(\underbrace{\text{proj}^1_0}_{:\mathbb{N}\to\mathbb{N}}, \underbrace{\underbrace{\text{succ}}_{:\mathbb{N}\to\mathbb{N}} \circ \underbrace{\text{proj}^3_2}_{:\mathbb{N}^3\to\mathbb{N}}}_{:\mathbb{N}^3\to\mathbb{N}})}_{:\mathbb{N}^2\to\mathbb{N}} : \mathbb{N}^2 \to \mathbb{N} \text{ is prim. rec.}$$
  (= addition)
- $$\underbrace{\text{primrec}(\underbrace{0}_{\in\mathbb{N}}, \underbrace{\text{proj}^2_0}_{:\mathbb{N}^2\to\mathbb{N}})}_{:\mathbb{N}^1\to\mathbb{N}} : \mathbb{N} \to \mathbb{N} \text{ is prim. rec.}$$
  (= pred)

Primitive Rec. Relations and Sets

- 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

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

  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

and to express

- $f(\vec{x}, y+1)$ 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:

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

(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:

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

  (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

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

Constant Function

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

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

Addition

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

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

Addition

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

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

Addition

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

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

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

Addition

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

Therefore

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

Multiplication

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

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

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

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

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

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

Multiplication

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

Therefore

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

Predecessor Function

- pred is prim. rec.:

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

Subtraction

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

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

Signum Function

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

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

is prim. rec.:

sig$(x) = x \dot{-} (x \dot{-} 1)$:

- For $x = 0$ we have

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

- For $x > 0$ we have

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

Signum Function

- Note that

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

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

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

$x < y$ is Prim. Rec.

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

- If $x < y$, then

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

  therefore

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

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

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

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

Add., Mult., Exp.

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

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

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

$$
x + y = ((+)\ 1)^y(x) \quad .
$$

Remark on Notation

- The notation $((+)\ 1)^y(x)$ is to be understood as follows:
  - Let f be a function (e.g. $((+)\ 1)$). Then we define

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

  - This is not to be confused with exponentiation

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

- So

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

Add., Mult., Exp.

- **Multiplication:**

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

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

$$x \cdot y = ((+)\ x)^y(0) \quad .$$

Add., Mult., Exp.

- **Exponentiation:**

$$x^0 = 1 \ ,$$
$$x^{y+1} = (x^y) \cdot x \ ,$$

Therefore, if we write $((\cdot)\ x)$ for the function $\mathbb{N} \to \mathbb{N}$, $((\cdot)\ x)(y) = x \cdot y$, then

$$x^y = ((\cdot)\ x)^y(1) \ .$$

- Note that above, we have both occurrences of x^y for exponentation and of $((\cdot)\ x)^y(1)$ for iterated function application.

Superexponentiation

- Extend this sequence further, by defining
  - **Superexponentiation:**

$$\text{superexp}(x, 0) = 1 \ ,$$
$$\text{superexp}(x, y+1) = x^{\text{superexp}(x,y)} \ ,$$

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

$$\text{superexp}(x, y) = ((\uparrow)\ x)^y(1) \ .$$

Supersuperexponentiation

- **Supersuperexponentiation:**

$$\begin{aligned}\mathrm{supersuperexp}(x,0) &= 1 \ ,\\ \mathrm{supersuperexp}(x,y+1) &= \mathrm{superexp}(x,\mathrm{supersuperexp}(x,y))\end{aligned}$$

- 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 Sect. 6 (a).

(b) Closure of the Prim. Rec. Func.

Closure under $\vee$, $\wedge$, $\neg$

- If $R, S \subseteq \mathbb{N}^n$ are prim. rec., so are
  - $R \vee S$,
  - $R \wedge S$,
  - $\neg R$.

Closure under Prop. Connectives

- Here
  - $(R \vee S)(\vec{x}) \Leftrightarrow R(\vec{x}) \vee S(\vec{x})$,
  - $(R \wedge S)(\vec{x}) \Leftrightarrow R(\vec{x}) \wedge S(\vec{x})$,
  - $(\neg R)(\vec{x}) \Leftrightarrow \neg R(\vec{x})$.
- 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:
    - $x \leq y \Leftrightarrow \neg(y < x)$.
    - $x = y \Leftrightarrow x \leq y \wedge y \leq x$.

Remark $\wedge$, $\vee$, $\mathbb{N}^n\setminus$

- We have
  - $R \vee S = R \cup S$ (the set theoretic union of R and S)
  - $R \wedge S = R \cap S$,
  - $\neg R = \mathbb{N}^n \setminus R$.

Closure under $\vee$, $\wedge$, $\neg$

- Proof of $R \cup S = R \vee S$:

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

Jump over Rest

- Proof of $R \cap S = R \wedge S$:

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

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

- Proof of $\mathbb{N}^n \setminus R = \neg R$:

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

Proof of Closure under $\vee$

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

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

Proof of Closure under $\vee$

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

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

Proof of Closure under $\vee$

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

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

Proof of Closure under $\wedge$

- $\chi_{R\wedge S}(\vec{x}) = \chi_R(\vec{x}) \cdot \chi_S(\vec{x}))$
  (and therefore $R \wedge S$ is primitive recursive):
  Jump over Rest of Proof
  - If $R(\vec{x})$ and $S(\vec{x})$ hold, then

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

Proof of Closure under $\wedge$

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

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

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

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

Proof of Closure under $\neg$

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

$$\underbrace{1 \dot{-} \underbrace{\chi_R(\vec{x})}_{=1}}_{=0} = 1 = \chi_{\neg R}(\vec{x}) \enspace .$$

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

$$\underbrace{1 \dot{-} \underbrace{\chi_R(\vec{x})}_{=0}}_{=1} = 1 = \chi_{\neg R}(\vec{x}) \enspace .$$

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

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

$$f(\vec{x}) = g_1(\vec{x}) \cdot \chi_R(\vec{x}) + g_2(\vec{x}) \cdot \chi_{\neg R}(\vec{x}) \quad \text{prim. rec.} \quad :$$

Jump over rest of proof.

- If $R(\vec{x})$ holds, then $\chi_R(\vec{x}) = 1$, $\chi_{negR}(\vec{x}) = 0$, therefore

$$\underbrace{\underbrace{g_1(\vec{x}) \cdot \underbrace{\chi_R(\vec{x})}_{=1}}_{=g_1(\vec{x})} + \underbrace{g_2(\vec{x}) \cdot \underbrace{\chi_{\neg R}(\vec{x})}_{=0}}_{=0}}_{=g_1(\vec{x})} = g_1(\vec{x}) = f(\vec{x}) \quad .$$

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

$$f(\vec{x}) = g_1(\vec{x}) \cdot \chi_R(\vec{x}) + g_2(\vec{x}) \cdot \chi_{\neg R}(\vec{x}) \quad :$$

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

$$\underbrace{\underbrace{g_1(\vec{x}) \cdot \underbrace{\chi_R(\vec{x})}_{=0}}_{=0} + \underbrace{g_2(\vec{x}) \cdot \underbrace{\chi_{\neg R}(\vec{x})}_{=1}}_{=g_2(\vec{x})}}_{=g_2(\vec{x})} = g_2(\vec{x}) = f(\vec{x}) \quad .$$

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

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

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 proofThe last equations follows from

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

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

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

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

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 and Example Factorial Function

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.:

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

Here, the last equations follows by

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

Jump over next Example

Example

Example for closure under bounded products:

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

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

$(f(0) = 0! = 1)$,
is primitive recursive, since

$$f(x) = \prod_{i<x} (i+1) = \prod_{i<x} g(i) \quad ,$$

where $g(y) := y + 1$ is prim. rec..
(Note that in the special case $x = 0$ we have

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

Remark on Factorial Function

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

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

Bounded Quantification

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

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

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

$$\prod_{z<y} \chi_R(\vec{x}, y) = \prod_{z<y} 1 = 1 = \chi_{R_1}(\vec{x}, y) \quad .$$

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

$$\prod_{z<y} \chi_R(\vec{x}, z) = 0 = \chi_{R_1}(\vec{x}, y) \quad .$$

Bounded Quantification

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

Proof for R_2:

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

Jump over Rest of Proof

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

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

Bounded Quantification

$R_2(\vec{x}, y) :\Leftrightarrow \exists z < y.R(\vec{x}, z)$.
Show $\chi_{R_2}(\vec{x}, y) = \mathrm{sig}(\sum_{z<y} \chi_R(\vec{x}, z))$

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

$$\sum_{z<y} \chi_R(\vec{x}, y) \geq \chi_R(\vec{x}, z) = 1 \quad ,$$

therefore

$$\mathrm{sig}(\sum_{z<y} \chi_R(\vec{x}, y)) = 1 = \chi_{R_2}(\vec{x}, y) \quad .$$

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

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

Bounded Search

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

- f can be defined by primitive recursion directly using the equations:

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

- Exercise: Show
  - f fulfills those equations
  - From these equations it follows that f is primitive recursive, provided R is.

Jump over Alternative Proof

Bounded Search

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

Alternative Proof of Closure under Bounded Search

Define

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

Q and Q' are primitive recursive.

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

We show

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

Jump over details.

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

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

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

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

Omit 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

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

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

Lemma 5.1

The coding and decoding functions for pairs, tuples and sequences of natural numbers are primitive recursive.

More precisely, the following functions are primitive recursive:

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

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

(c) $\pi^k : \mathbb{N}^k \rightarrow \mathbb{N}$ ($k \geq 1$).
(Remember $\pi^k(x_0, \ldots, x_{k-1})$ encodes the sequence $(x_0, \ldots, x_{k-1})$.

Lemma 5.1

(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 $\pi_i^k(\pi^k(x_0, \ldots, x_{k-1})) = x_i$ 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) $\mathrm{lh} : \mathbb{N} \to \mathbb{N}$.
(Remember that $\mathrm{lh}(\langle x_0, \ldots, x_{k-1} \rangle) = k$.)

Lemma 5.1

(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 5.1 (a), (b)

(a)

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

is primitive recursive.

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

$$\begin{aligned}\pi_0(x) &:= \mu y < x+1.\exists z < x+1.x = \pi(y,z) \quad , \\ \pi_1(x) &:= \mu z < x+1.\exists y < x+1.x = \pi(y,z) \quad .\end{aligned}$$

Therefore π_0, π_1 are primitive recursive.

Proof of Lemma 5.1 (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 5.1 (d)

(d) We have

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

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 $g : \mathbb{N}^2 \to \mathbb{N}$,

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

which is primitive recursive.

Proof of Lemma 5.1 (d)

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

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

So f is primitive recursive.

Proof of Lemma 5.1 (e), (f), (g)

(e)

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

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 5.2

(Technical Lemma needed in the proof of closure under course-of-value primitive recursion below.)

Prim. rec. functions as follows do exist:

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

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

Jump over proof.

Proof of Lemma 5.2 (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 5.2 (a)

We have

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

Proof of Lemma 5.2 (b)

Proof for $\mathsf{beginning}$**:**
Define

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

Proof of Lemma 5.2 (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 5.2 (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 5.2 (b)

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

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

Then

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

Proof of Lemma 5.2 (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 5.2 (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

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

Definition Course-Of-Value

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

$$\begin{aligned} \overline{f} &: \mathbb{N}^{n+1} \to \mathbb{N} \\ \overline{f}(\vec{x}, n) &:= \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, n-1) \rangle \end{aligned}$$

  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

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

is primitive recursive.
Then

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

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

is prim. rec.

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.

Example

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

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

Definable by course-of-value primitive recursion:

- We have

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

using $(\overline{\mathsf{fib}}(x))_{x-2} = \mathsf{fib}(x-2)$, $(\overline{\mathsf{fib}}(x))_{x-1} = \mathsf{fib}(x-1)$.

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

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

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

Therefore $\overline{f}$ is primitive recursive.

Proof

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

Now we have that

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

is primitive recursive.

Lemma and Definition 5.3

(Technical Lemma used later to simulate Turing Machines using primitive recursive/partial recursive functions).

There exist prim. rec. functions as follows:

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

$$\mathsf{append}(\langle x_0, \ldots, x_{k-1}\rangle, \langle y_0, \ldots, y_{l-1}\rangle) = \langle x_0, \ldots, x_{k-1}, y_0, \ldots, y_{l-1}\rangle \ .$$

We write $x * y$ for $\mathsf{append}(x, y)$.

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

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

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 5.3

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

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

and if $i \geq n$,

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

Lemma and Definition 5.3

(d) $\mathsf{half} : \mathbb{N} \to \mathbb{N}$,
s.t. $\mathsf{half}(x) = y$ if $x = 2y$ or $x = 2y + 1$.

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

(f) A function $\mathsf{bin}^{-1} : \mathbb{N} \to \mathbb{N}$, s.t.
$\mathsf{bin}^{-1}(\langle b_0, \ldots, b_k \rangle) = x$, if $(b_0, \ldots, b_k)_2 = x$.

The proof will be omitted in the lecture.

Jump over proof.

Proof of Lemma 5.3 (a)

We have

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

Proof of Lemma 5.3 (a)

Therefore we have

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

One can see that $\mathsf{beginning}(x) < x$ 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 5.3 (b)

We have

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

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

Proof of Lemma 5.3 (c)

We can define

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

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

Proof of Lemma 5.3 (d), (e)

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

(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 5.3 (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.