

5. The Primitive Recursive Function

- 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.

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

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 recursive 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**.
- 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.

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

Inductive definition of the primitive recursive functions

$$f : \mathbb{N}^k \rightarrow \mathbb{N}.$$

- The following basic Functions are primitive recursive:
 - zero : $\mathbb{N} \rightarrow \mathbb{N}$,
 - succ : $\mathbb{N} \rightarrow \mathbb{N}$,
 - $\text{proj}_i^k : \mathbb{N}^k \rightarrow \mathbb{N}$ ($0 \leq i < k$).

Remember that these functions have defining equations

- $\text{zero}(n) = 0$,
- $\text{succ}(n) = n + 1$,
- $\text{proj}_i^k(a_0, \dots, a_{k-1}) = a_i$.

Def. Prim. Rec. Functions

• If

• $f : \mathbb{N}^k \rightarrow \mathbb{N}$ is primitive recursive,

• $g_i : \mathbb{N}^n \rightarrow \mathbb{N}$ are primitive recursive, ($i = 0, \dots, k - 1$),

so is

$$f \circ (g_0, \dots, g_{k-1}) : \mathbb{N}^n \rightarrow \mathbb{N} .$$

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

$$h(\vec{x}) = f(g_0(\vec{x}), \dots, g_{k-1}(\vec{x})) .$$

Especially, if $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$ are primitive recursive, so is

$$f \circ g : \mathbb{N} \rightarrow \mathbb{N} .$$

Def. Prim. Rec. Functions

- If
 - $g : \mathbb{N}^n \rightarrow \mathbb{N}$,
 - $h : \mathbb{N}^{n+2} \rightarrow \mathbb{N}$ are primitive recursive,so is the function $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ defined by primitive recursion from g, h .
- Remember that f is defined by
 - $f(\vec{x}, 0) = g(\vec{x})$,
 - $f(\vec{x}, n + 1) = h(\vec{x}, n, f(\vec{x}, n))$.
- f is denoted by $\text{primrec}(g, h)$.

Def. Prim. Rec. Functions

- If
 - $k \in \mathbb{N}$,
 - $h : \mathbb{N}^2 \rightarrow \mathbb{N}$ is primitive recursive,
so is the function $f : \mathbb{N} \rightarrow \mathbb{N}$, defined by primitive recursion from k and h .
- Remember that $f := \text{primrec}(k, h)$ is defined by
 - $f(0) = k$,
 - $f(n + 1) = h(n, f(n))$.
- f is denoted by $\text{primrec}(k, h)$.

Inductively Defined Sets

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

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

Inductively Defined Sets

E.g.

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

(= addition)

$$\bullet \text{ primrec}(\underbrace{0}_{\in \mathbb{N}}, \underbrace{\text{proj}_0^2}_{:\mathbb{N}^2 \rightarrow \mathbb{N}}) : \mathbb{N} \rightarrow \mathbb{N} \text{ is prim. rec.}$$

(= pred)

Primitive Rec. Relations and Sets

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

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

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

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

Identity Function

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

Constant Function

- $\text{const}_n : \mathbb{N} \rightarrow \mathbb{N}$, $\text{const}_n(k) = n$ is primitive recursive:

$$\text{const}_n = \underbrace{\text{succ} \circ \dots \circ \text{succ}}_{n \text{ times}} \circ \text{zero}:$$

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

Addition

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

$$\text{add}(x, 0) = x + 0$$

$$= x$$

$$\text{add}(x, y + 1) = x + (y + 1)$$

$$= (x + y) + 1$$

$$= \text{add}(x, y) + 1$$

$$= \text{succ}(\text{add}(x, y))$$

Addition

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

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

Addition

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

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

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

Addition

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

Therefore

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

Multiplication

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

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

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

Multiplication

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

• $\text{mult}(x, y + 1) = h(x, y, \text{mult}(x, y))$,
where

$$h : \mathbb{N}^3 \rightarrow \mathbb{N}, h(x, y, z) := \text{add}(z, x).$$

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

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

Multiplication

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

Therefore

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

Predecessor Function

- pred is prim. rec.:

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

Subtraction

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

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

Signum Function

• $\text{sig} : \mathbb{N} \rightarrow \mathbb{N}$,

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

is prim. rec.:

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

• 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

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

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

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

$x < y$ is Prim. Rec.

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

• If $x < y$, then

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

therefore

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

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

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

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

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 $\mathbb{N} \rightarrow \mathbb{N}$,
 $((+) 1)(n) = n + 1$, then

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

Remark on Notation

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

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

- This is not to be confused with exponentiation

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

- So

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

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 $\mathbb{N} \rightarrow \mathbb{N}$, $((+) n)(k) = k + n$, then

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

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 $\mathbb{N} \rightarrow \mathbb{N}$,
 $((\cdot) n)(m) = n \cdot m$, then

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

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

Superexponentiation

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

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

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

$$\text{superexp}(n, m) = ((\uparrow) n)^m(1).$$

Supersuperexponentiation

- **Supersuperexponentiation:**

$$\text{supersuperexp}(n, 0) = 1 ,$$

$$\text{supersuperexp}(n, m + 1) = \text{superexp}(n, \text{supersuperexp}(n, m))$$

- Etc.

- One obtains sequence of extremely fast growing functions.
- These functions will exhaust the primitive recursive functions.
- We will reconsider this sequence at the beginning of Sect. 6 (a).

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

Closure under \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

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

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

Proof of Closure under \vee

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

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

$\underbrace{\hspace{10em}}_{\geq 1}$

$\underbrace{\hspace{10em}}_{=1}$

Proof of Closure under \vee

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

$$\text{sig}\left(\underbrace{\chi_R(\vec{x})}_{=0} + \underbrace{\chi_S(\vec{x})}_{=0}\right) = 0 = \chi_{R \vee S}(\vec{x}) \ .$$
$$\underbrace{\hspace{10em}}_{=0}$$

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

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

Proof of Closure under \wedge

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

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

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

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

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

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

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

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

Definition by Cases

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

Assume

- $g_1, g_2 : \mathbb{N}^n \rightarrow \mathbb{N}$ are primitive recursive,
- $R \subseteq \mathbb{N}^n$ is primitive recursive.

Then $f : \mathbb{N}^n \rightarrow \mathbb{N}$,

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

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 $\chi_R(\vec{x}) = 1$,
 $\chi_{\neg R}(\vec{x}) = 0$, therefore

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

Definition by Cases

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

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

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

where

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

and for $y > 0$,

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

Bounded Sums

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

Proof that f is prim. rec.:

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

Jump over rest of proof The last equations follows from

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

Example

- We have above

$$f(\vec{x}, 0) = g(\vec{x}, 0)$$

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

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

etc.

Bounded Products

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

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

where

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

and for $y > 0$,

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

Omit Proof and Example Factorial Function

Bounded Products

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

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

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

[Jump over next Example](#)

Example

Example for closure under bounded products:

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

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

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

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

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

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

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

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

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

Bounded Quantification

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

Proof for R_1 :

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

Jump over details.

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

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

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

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

Bounded Quantification

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

Proof for R_2 :

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

Jump over Rest of Proof

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

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

Bounded Quantification

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

$$\text{Show } \chi_{R_2}(\vec{x}, y) = \text{sig}\left(\sum_{z < y} \chi_R(\vec{x}, z)\right)$$

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

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

therefore

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

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:

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

- 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

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

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

Q and Q' are primitive recursive.

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

We show

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

Jump over details.

Bounded Search

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

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

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

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

Let z be minimal s.t. $R(\vec{x}, z)$.

$$\Rightarrow Q(\vec{x}, z),$$

$$\Rightarrow \chi_Q(\vec{x}, z) \cdot z = z \quad .$$

For $z \neq z'$ we have $\neg Q(\vec{x}, z')$,

therefore $\chi_Q(\vec{x}, z') \cdot z' = 0$ ($z' \neq z$).

Furthermore, $\neg Q'(\vec{x}, y)$, therefore $\chi_{Q'}(\vec{x}, y) \cdot y = 0$.

Therefore

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

Bounded Search

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

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

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

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

$$\Rightarrow \neg Q(\vec{x}, z) \text{ for } z < y,$$

$$\Rightarrow \forall z < y. \chi_Q(\vec{x}, z) \cdot z = 0.$$

Furthermore, $Q'(\vec{x}, y)$,

therefore $\chi_{Q'}(\vec{x}, y) \cdot y = y$.

Therefore

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

Example

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

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

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

$$f(x) = \sum_{y < x} \chi_P(y) .$$

Example 2

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

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

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, $\pi(n, m)$ encodes two natural numbers as one.)
- (b) $\pi_0, \pi_1 : \mathbb{N} \rightarrow \mathbb{N}$.
(Remember $\pi_0(\pi(n, m)) = n$, $\pi_1(\pi(n, m)) = m$).
- (c) $\pi^k : \mathbb{N}^k \rightarrow \mathbb{N}$ ($k \geq 1$).
(Remember $\pi^k(n_0, \dots, n_{k-1})$ encodes the sequence (n_0, \dots, n_k)).

Lemma 5.1

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

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

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

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

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

$$f_k(x_0, \dots, x_{k-1}) = \langle x_0, \dots, x_{k-1} \rangle.$$

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

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

(Remember that $\text{lh}(\langle x_0, \dots, x_{k-1} \rangle) = k$.)

Lemma 5.1

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

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

The proof will be omitted in the lecture.

Jump over proof.

Proof of Lemma 5.1 (a), (b)

(a)

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

is primitive recursive.

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

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

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

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

Therefore π^{k+1} is primitive recursive
(using that π , π^k are primitive recursive).

Proof of Lemma 5.1 (d)

(d) We have

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

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 5.1 (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 $g : \mathbb{N}^2 \rightarrow \mathbb{N}$,

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

which is primitive recursive.

Proof of Lemma 5.1 (d)

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

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

So f is primitive recursive.

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

(e)

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

is primitive recursive.

(f)

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

(g)

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

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

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

- **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) $\text{last} : \mathbb{N} \rightarrow \mathbb{N}$ and $\text{beginning} : \mathbb{N} \rightarrow \mathbb{N}$ s.t.

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

[Jump over proof.](#)

Proof of Lemma 5.2 (a)

Define

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

so snoc is primitive recursive.

Proof of Lemma 5.2 (a)

We have

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

Proof of Lemma 5.2 (b)

Proof for beginning:

Define

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

Proof of Lemma 5.2 (b)

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

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

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

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

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

Proof of Lemma 5.2 (b)

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

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

Proof of Lemma 5.2 (b)

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

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

Then

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

Proof of Lemma 5.2 (b)

Therefore

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

Proof of Lemma 5.2 (b)

Proof for last:

Define

$$\text{last}(x) := (x)_{\text{lh}(x) \div 1}$$

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

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

Definition Course-Of-Value

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

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

$$\bar{f}(\vec{x}, n) := \langle f(\vec{x}, 0), f(\vec{x}, 1), \dots, f(\vec{x}, n - 1) \rangle$$

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

- \bar{f} is called the course-of-value function associated with f .

Course-of-Value Prim. Recursion

The prim. rec. functions are closed under

course-of-value primitive recursion:

Assume

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

is primitive recursive.

Then

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

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

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.

$\text{fib} : \mathbb{N} \rightarrow \mathbb{N}$ given by:

$$\text{fib}(0) := 1 ,$$

$$\text{fib}(1) := 1 ,$$

$$\text{fib}(n) := \text{fib}(n - 1) + \text{fib}(n - 2), \text{ if } n > 1,$$

Definable by course-of-value primitive recursion:

• We have

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

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

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

$$\bar{f}(\vec{x}, 0) = \langle \rangle ,$$

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

Therefore \bar{f} is primitive recursive.

Proof

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

Now we have that

$$\begin{aligned} f(\vec{x}, y) &= (\langle f(\vec{x}, 0), \dots, f(\vec{x}, y) \rangle)_y \\ &= (\bar{f}(\vec{x}, y + 1))_y \\ &= \text{last}(\bar{f}(\vec{x}, y + 1)) \end{aligned}$$

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) $\text{append} : \mathbb{N}^2 \rightarrow \mathbb{N}$ s.t.

$$\begin{aligned} \text{append}(\langle n_0, \dots, n_{k-1} \rangle, \langle m_0, \dots, m_{l-1} \rangle) \\ = \langle n_0, \dots, n_{k-1}, m_0, \dots, m_{l-1} \rangle . \end{aligned}$$

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

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

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

and if $i \geq n$, then

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

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

Lemma and Definition 5.3

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

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

and if $i \geq n$,

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

Lemma and Definition 5.3

- (d) $\text{half} : \mathbb{N} \rightarrow \mathbb{N}$,
s.t. $\text{half}(n) = k$ if $n = 2k$ or $n = 2k + 1$.
- (e) The function $\text{bin} : \mathbb{N} \rightarrow \mathbb{N}$, s.t.
 $\text{bin}(n) = \langle b_0, \dots, b_k \rangle$,
for b_i in normal form (no leading zeros, unless $n = 0$),
s.t. $n = (b_0, \dots, b_k)_2$
- (f) A function $\text{bin}^{-1} : \mathbb{N} \rightarrow \mathbb{N}$, s.t.
 $\text{bin}^{-1}(\langle b_0, \dots, b_k \rangle) = n$, if $(b_0, \dots, b_k)_2 = n$.

The proof will be omitted in the lecture.

Jump over proof.

Proof of Lemma 5.3 (a)

We have

$$\begin{aligned} & \text{append}(\langle x_0, \dots, x_n \rangle, 0) \\ &= \text{append}(\langle x_0, \dots, x_n \rangle, \langle \rangle) \\ &= \langle x_0, \dots, x_n \rangle, \end{aligned}$$

and for $m > 0$

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

Proof of Lemma 5.3 (a)

Therefore we have

$$\text{append}(x, 0) = x ,$$

$$\text{append}(x, y) = \text{snoc}(\text{append}(x, \text{beginning}(y)), \text{last}(y)) ,$$

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

Proof of Lemma 5.3 (b)

We have

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

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

Proof of Lemma 5.3 (c)

We can define

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

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

Proof of Lemma 5.3 (d), (e)

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

(e)

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

therefore definable by course-of-value primitive recursion.

Proof of Lemma 5.3 (f)

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

therefore definable by course-of-value primitive recursion.