



Computability Theory

Aart Middeldorp

Initial Remarks

- ▶ **Computability Theory** is part of WM 9 in master program Computer Science

Initial Remarks

- ▶ **Computability Theory** is part of WM 9 in master program Computer Science
- ▶ WM 9 is part of **Logic and Learning** specialization

Initial Remarks

- ▶ **Computability Theory** is part of WM 9 in master program Computer Science
- ▶ WM 9 is part of **Logic and Learning** specialization
- ▶ other master modules with theory content:
 - ▶ Program and Resource Analysis (WM 8)
 - ▶ Advanced Functional Programming (WM 20)
 - ▶ **The Lambda-Calculus, Typed and Untyped** (WM 20)

Outline

- 1. Organisation**
- 2. Contents**
- 3. Primitive Recursive Functions**
- 4. Primitive Recursive Predicates**
- 5. Pairing**
- 6. Summary**

- ▶ LVA 703317

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS
- ▶ 13:45–16:15 in HS 11

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS
- ▶ 13:45–16:15 in HS 11
- ▶ <https://cl-teaching.uibk.ac.at/26w/ct>

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS
- ▶ 13:45–16:15 in HS 11
- ▶ <https://cl-teaching.uibk.ac.at/26w/ct>
- ▶ OLAT

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS
- ▶ 13:45–16:15 in HS 11
- ▶ <https://cl-teaching.uibk.ac.at/26w/ct>
- ▶ OLAT
- ▶ consultation hours: Monday 11:30–13:00 in 3M07

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS
- ▶ 13:45–16:15 in HS 11
- ▶ <https://cl-teaching.uibk.ac.at/26w/ct>
- ▶ OLAT
- ▶ consultation hours: Monday 11:30–13:00 in 3M07

Schedule

lecture 1	October 5	lecture 5	November 16	lecture 9	December 14
lecture 2	October 12	lecture 6	November 23	lecture 10	January 11
lecture 3	October 19	lecture 7	November 30	lecture 11	January 18
lecture 4	November 9	lecture 8	December 7	lecture 12	January 25

Organisation

- ▶ LVA 703317
- ▶ VU 3 – 5 ECTS
- ▶ 13:45–16:15 in HS 11
- ▶ <https://cl-teaching.uibk.ac.at/26w/ct>
- ▶ OLAT
- ▶ consultation hours: Monday 11:30–13:00 in 3M07

Schedule

lecture 1	October 5	lecture 5	November 16	lecture 9	December 14
lecture 2	October 12	lecture 6	November 23	lecture 10	January 11
lecture 3	October 19	lecture 7	November 30	lecture 11	January 18
lecture 4	November 9	lecture 8	December 7	lecture 12	January 25 (test)

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

$$\text{score} = \min \left(\max \left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B \right), 100 \right)$$

E : points for solved **exercises** (at most 90)

Grading

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved exercises (at most 90)

P : points for **presentation** of solutions (at most 10)

Grading

$$\text{score} = \min \left(\max \left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B \right), 100 \right)$$

E : points for solved exercises (at most 90)

P : points for presentation of solutions (at most 10)

T : points for **test** (at most 100)

Grading

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved exercises (at most 90)

P : points for presentation of solutions (at most 10)

T : points for test (at most 100)

B : points for **bonus exercises** (at most 15)

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved exercises (at most 90)

P : points for presentation of solutions (at most 10)

T : points for test (at most 100)

B : points for bonus exercises (at most 15)

$$\text{grade} = \text{score} \in (-50) \rightarrow 5 \quad [50 - 63) \rightarrow 4 \quad [63 - 75) \rightarrow 3 \quad [75 - 88) \rightarrow 2 \quad [88 -) \rightarrow 1$$

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved **exercises** (at most 90)

P : points for presentation of solutions (at most 10)

T : points for test (at most 100)

B : points for **bonus exercises** (at most 15)

$$\text{grade} = \text{score} \in (-50) \rightarrow 5 \quad [50 - 63) \rightarrow 4 \quad [63 - 75) \rightarrow 3 \quad [75 - 88) \rightarrow 2 \quad [88 -) \rightarrow 1$$

- ▶ solved exercises must be marked and solutions (**PDF format**) must be uploaded in **OLAT** before **10 am on Monday**

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved exercises (at most 90)

P : points for **presentation** of solutions (at most 10)

T : points for test (at most 100)

B : points for bonus exercises (at most 15)

$$\text{grade} = \text{score} \in (-50) \rightarrow 5 \quad [50 - 63) \rightarrow 4 \quad [63 - 75) \rightarrow 3 \quad [75 - 88) \rightarrow 2 \quad [88 -) \rightarrow 1$$

- ▶ solved exercises must be marked and solutions (PDF format) must be uploaded in OLAT before 10 am on Monday
- ▶ presentations are optional

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved exercises (at most 90)

P : points for presentation of solutions (at most 10)

T : points for **test** (at most 100)

B : points for bonus exercises (at most 15)

$$\text{grade} = \text{score} \in (-50) \rightarrow 5 \quad [50 - 63) \rightarrow 4 \quad [63 - 75) \rightarrow 3 \quad [75 - 88) \rightarrow 2 \quad [88 -) \rightarrow 1$$

- ▶ solved exercises must be marked and solutions (PDF format) must be uploaded in OLAT before 10 am on Monday
- ▶ presentations are optional
- ▶ (optional) test on January 25

$$\text{score} = \min\left(\max\left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B\right), 100\right)$$

E : points for solved exercises (at most 90)

P : points for presentation of solutions (at most 10)

T : points for test (at most 100)

B : points for bonus exercises (at most 15)

$$\text{grade} = \text{score} \in (-50) \rightarrow 5 \quad [50 - 63) \rightarrow 4 \quad [63 - 75) \rightarrow 3 \quad [75 - 88) \rightarrow 2 \quad [88 -) \rightarrow 1$$

- ▶ solved exercises must be marked and solutions (PDF format) must be uploaded in OLAT before 10 am on Monday
- ▶ presentations are optional
- ▶ (optional) test on January 25

evaluation 23W

Outline

1. Organisation

AI Policy

2. Contents

3. Primitive Recursive Functions

4. Primitive Recursive Predicates

5. Pairing

6. Summary

- ▶ students are responsible for all content they deliver (spoken, in writing, or code)

AI Policy — Principles

- ▶ students are responsible for all content they deliver (spoken, in writing, or code)
- ▶ **traffic light** system recognizes that policies differ between courses; its purpose is to provide students with certainty

AI Policy — Principles

- ▶ students are responsible for all content they deliver (spoken, in writing, or code)
- ▶ **traffic light** system recognizes that policies differ between courses; its purpose is to provide students with certainty



AI use is prohibited



AI use is allowed



AI use is permitted, but must be declared and comply with other restrictions (specific details to be provided by course teacher)



AI use is recommended

AI Policy — Principles

- ▶ students are responsible for all content they deliver (spoken, in writing, or code)
- ▶ traffic light system recognizes that policies differ between courses; its purpose is to provide students with certainty
- ▶ teachers have authority to redefine any aspect of policy and its enforcement



AI use is prohibited



AI use is allowed



AI use is permitted, but must be declared and comply with other restrictions (specific details to be provided by course teacher)



AI use is recommended

AI Policy — Principles

- ▶ students are responsible for all content they deliver (spoken, in writing, or code)
- ▶ traffic light system recognizes that policies differ between courses; its purpose is to provide students with certainty
- ▶ teachers have authority to redefine any aspect of policy and its enforcement
- ▶ failure to comply with policy may result in failing course



AI use is prohibited



AI use is allowed



AI use is permitted, but must be declared and comply with other restrictions (specific details to be provided by course teacher)



AI use is recommended

AI Policy — Principles

- ▶ students are responsible for all content they deliver (spoken, in writing, or code)
- ▶ traffic light system recognizes that policies differ between courses; its purpose is to provide students with certainty
- ▶ teachers have authority to redefine any aspect of policy and its enforcement
- ▶ failure to comply with policy may result in failing course



AI use is prohibited



AI use is allowed



AI use is permitted, but must be declared and comply with other restrictions (specific details to be provided by course teacher)



AI use is recommended



AI use is permitted with restrictions



AI use is permitted with restrictions

ideation

data analysis

text generation

generation of solution hints

proof generation

proof verification

creation of presentation slides

literature search

programming

production of multimedia content

text summarisation

spell checking

improvement of graphical elements

text translation

text editing



AI use is permitted with restrictions

- | | | |
|-----------------------------------|--------------------|--------------------|
| ideation | data analysis | text generation |
| generation of solution hints | proof generation | proof verification |
| creation of presentation slides | literature search | programming |
| production of multimedia content | text summarisation | spell checking |
| improvement of graphical elements | text translation | text editing |

Outline

1. Organisation

2. Contents

3. Primitive Recursive Functions

4. Primitive Recursive Predicates

5. Pairing

6. Summary

- ▶ Turing machines

- ▶ lambda calculus
- ▶ Turing machines

Theorem

- ▶ combinatory logic
- ▶ lambda calculus
- ▶ Turing machines
- ▶ term rewrite systems

Theorem

- ▶ combinatory logic
- ▶ lambda calculus
- ▶ Turing machines
- ▶ C/Haskell/Java/Python/... programs
- ▶ term rewrite systems

Theorem

- ▶ combinatory logic
- ▶ enumeration machines
- ▶ interaction nets
- ▶ lambda calculus
- ▶ Turing machines
- ▶ C/Haskell/Java/Python/... programs
- ▶ recursive functions
- ▶ term rewrite systems
- ▶ two-counter automata
- ▶ quantum computers

capture the same notion of **computation**

Theorem

- ▶ combinatory logic
- ▶ enumeration machines
- ▶ interaction nets
- ▶ lambda calculus
- ▶ Turing machines
- ▶ C/Haskell/Java/Python/... programs
- ▶ recursive functions
- ▶ term rewrite systems
- ▶ two-counter automata
- ▶ quantum computers

capture the same notion of computation

Literature (Recursive Function Theory)

- ▶ Nigel Cutland
Computability: An Introduction to Recursive Function Theory
Cambridge University Press, 1980
- ▶ Richard Epstein and Walter Carnielli
Computability: Computable Functions, Logic, and the Foundations of Mathematics (3rd edition)
Advanced Reasoning Forum, 2008
- ▶ Piergiorgio Odifreddi
Classical Recursion Theory (2nd edition)
North Holland, 1992
- ▶ Hartley Rogers Jr.
Theory of Recursive Functions and Effective Computability
MIT Press, 1987
- ▶ Rózsa Péter
Rekursive Funktionen in der Komputer-Theorie
Akadémiai Kiadó, 1976

Literature (Combinatory Logic)

- ▶ Katalin Bimbó
Combinatory Logic: Pure, Applied and Typed
CRC Press, 2011
- ▶ J. Roger Hindley and Jonathan P. Seldin
Lambda-Calculus and Combinators, an Introduction
Cambridge University Press, 2008

Part I: Recursive Function Theory

Ackermann function, bounded minimization, bounded recursion, course-of-values recursion, diagonalization, diophantine sets, elementary functions, fixed point theorem, Fibonacci numbers, Gödel numbering, Gödel's β function, Grzegorczyk hierarchy, loop programs, minimization, normal form theorem, partial recursive functions, primitive recursion, recursive enumerability, recursive inseparability, s-m-n theorem, total recursive functions, undecidability, while programs

Part II: Combinatory Logic

abstraction, arithmetization, CL-representability, combinators, combinatorial completeness, Church numerals, Church-Rosser theorem, Curry-Howard isomorphism, fixed point theorem, intuitionistic propositional logic, normalization theorem, termination, typing, undecidability, Z property

Part I: Recursive Function Theory

Ackermann function, **bounded minimization**, bounded recursion, **course-of-values recursion**, diagonalization, diophantine sets, elementary functions, fixed point theorem, Fibonacci numbers, Gödel numbering, Gödel's β function, Grzegorczyk hierarchy, loop programs, minimization, normal form theorem, partial recursive functions, **primitive recursion**, recursive enumerability, recursive inseparability, s-m-n theorem, total recursive functions, undecidability, while programs

Part II: Combinatory Logic

abstraction, arithmetization, CL-representability, combinators, combinatorial completeness, Church numerals, Church-Rosser theorem, Curry-Howard isomorphism, fixed point theorem, intuitionistic propositional logic, normalization theorem, termination, typing, undecidability, Z property

Outline

1. Organisation
2. Contents
- 3. Primitive Recursive Functions**
4. Primitive Recursive Predicates
5. Pairing
6. Summary

Examples

- ▶ which function is computable ?

$$f(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of exactly } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of at least } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

Examples

- ▶ which function is computable ?

$$f(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of exactly } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of at least } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

- ▶ which are (total) functions ?

$$f(x) = f(x) + 1$$

Examples

- ▶ which function is computable ?

$$f(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of exactly } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of at least } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

- ▶ which are (total) functions ?

$$f(x) = f(x) + 1$$

$$g(x) = \begin{cases} 0 & \text{if } x = 0 \\ g(g(x-1)) & \text{if } x > 0 \end{cases}$$

Examples

- ▶ which function is computable ?

$$f(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of exactly } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of at least } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

- ▶ which are (total) functions ?

$$f(x) = f(x) + 1$$

$$g(x) = \begin{cases} 0 & \text{if } x = 0 \\ g(g(x-1)) & \text{if } x > 0 \end{cases}$$

$$h(x) = 0 \times f(x)$$

Examples

- ▶ which function is computable ?

$$f(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of exactly } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

$$g(x) = \begin{cases} 1 & \text{if decimal expansion of } \pi \text{ contains consecutive run of at least } x \text{ fives} \\ 0 & \text{otherwise} \end{cases}$$

- ▶ which are (total) functions ?

$$f(x) = f(x) + 1$$

$$g(x) = \begin{cases} 0 & \text{if } x = 0 \\ g(g(x-1)) & \text{if } x > 0 \end{cases}$$

$$i(x) = \begin{cases} 1 & \text{if } x = 0 \text{ or } x = 1 \\ i(x/2) & \text{if } x > 1 \text{ is even} \\ i(3x+1) & \text{if } x > 1 \text{ is odd} \end{cases}$$

$$h(x) = 0 \times f(x)$$

Definition

class **PR** of **primitive recursive functions** is smallest class of total functions $f: \mathbb{N}^n \rightarrow \mathbb{N}$

Definition

class **PR** of primitive recursive functions is smallest class of total functions $f: \mathbb{N}^n \rightarrow \mathbb{N}$ that contains all **initial functions**

► **zero** $z(x) = 0$

Definition

class **PR** of primitive recursive functions is smallest class of total functions $f: \mathbb{N}^n \rightarrow \mathbb{N}$ that contains all **initial functions**

- ▶ zero $z(x) = 0$
- ▶ **successor** $s(x) = x + 1$

Definition

class **PR** of primitive recursive functions is smallest class of total functions $f: \mathbb{N}^n \rightarrow \mathbb{N}$ that contains all **initial functions**

- ▶ zero $z(x) = 0$
- ▶ successor $s(x) = x + 1$
- ▶ **projection** $\pi_i^n(x_1, \dots, x_n) = x_i$ for all $n \geq 1$ and $1 \leq i \leq n$

Definition

class **PR** of primitive recursive functions is smallest class of total functions $f: \mathbb{N}^n \rightarrow \mathbb{N}$ that contains all initial functions

- ▶ zero $z(x) = 0$
- ▶ successor $s(x) = x + 1$
- ▶ projection $\pi_i^n(x_1, \dots, x_n) = x_i$ for all $n \geq 1$ and $1 \leq i \leq n$

and is closed under **composition**

- ▶ $f(\vec{x}) = g(h_1(\vec{x}), \dots, h_m(\vec{x})) \in \text{PR}$ for all $g: \mathbb{N}^m \rightarrow \mathbb{N} \in \text{PR}$ and $h_1, \dots, h_m: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$

Definition

class **PR** of primitive recursive functions is smallest class of total functions $f: \mathbb{N}^n \rightarrow \mathbb{N}$ that contains all initial functions

- ▶ zero $z(x) = 0$
- ▶ successor $s(x) = x + 1$
- ▶ projection $\pi_i^n(x_1, \dots, x_n) = x_i$ for all $n \geq 1$ and $1 \leq i \leq n$

and is closed under composition

- ▶ $f(\vec{x}) = g(h_1(\vec{x}), \dots, h_m(\vec{x})) \in \text{PR}$ for all $g: \mathbb{N}^m \rightarrow \mathbb{N} \in \text{PR}$ and $h_1, \dots, h_m: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$

and **primitive recursion**

- ▶ $f(x, \vec{y}): \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ defined by

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

$$f(x+1, \vec{y}) = h(f(x, \vec{y}), x, \vec{y})$$

belongs to PR for all $g: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $h: \mathbb{N}^{n+2} \rightarrow \mathbb{N} \in \text{PR}$

Examples

► addition

$$x + y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

Examples

► addition

$$x + y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = \pi_1^1(x)$

Examples

► addition

$$x + y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = \pi_1^1(x)$ and $h(x, y, z) = s(\pi_1^3(x, y, z))$

Examples

► addition

$$x + y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = \pi_1^1(x)$ and $h(x, y, z) = s(\pi_1^3(x, y, z))$

► multiplication

$$x \times y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

Examples

► addition

$$x + y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = \pi_1^1(x)$ and $h(x, y, z) = s(\pi_1^3(x, y, z))$

► multiplication

$$x \times y = f(x, y) \in \text{PR}$$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = z(x)$

Examples

► addition $x + y = f(x, y) \in \text{PR}$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = \pi_1^1(x)$ and $h(x, y, z) = s(\pi_1^3(x, y, z))$

► multiplication $x \times y = f(x, y) \in \text{PR}$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = z(x)$ and $h(x, y, z) = \pi_1^3(x, y, z) + \pi_3^3(x, y, z)$

Examples

- addition $x + y = f(x, y) \in \text{PR}$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = \pi_1^1(x)$ and $h(x, y, z) = s(\pi_1^3(x, y, z))$

- multiplication $x \times y = f(x, y) \in \text{PR}$

$$f(0, y) = g(y)$$

$$f(x + 1, y) = h(f(x, y), x, y)$$

with $g(x) = z(x)$ and $h(x, y, z) = \pi_1^3(x, y, z) + \pi_3^3(x, y, z)$

- **exponentiation** $x^y = f(y, x) \in \text{PR}$

$$f(0, y) = s(z(y))$$

$$f(x + 1, y) = \pi_1^3(f(x, y), x, y) \times \pi_3^3(f(x, y), x, y)$$

Lemma

$g(x_1, \dots, x_m) = f(y_1, \dots, y_n) \in \text{PR}$ if $f: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $y_i \in \{x_1, \dots, x_m\}$ for all $1 \leq i \leq n$

Lemma

$g(x_1, \dots, x_m) = f(y_1, \dots, y_n) \in \text{PR}$ if $f: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $y_i \in \{x_1, \dots, x_m\}$ for all $1 \leq i \leq n$

Proof

$$\triangleright y_i = x_j \implies y_i = \pi_j^m(x_1, \dots, x_m)$$

Lemma

$g(x_1, \dots, x_m) = f(y_1, \dots, y_n) \in \text{PR}$ if $f: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $y_i \in \{x_1, \dots, x_m\}$ for all $1 \leq i \leq n$

Proof

- ▶ $y_i = x_j \implies y_i = \pi_j^m(x_1, \dots, x_m)$
- ▶ hence g can be defined by **composing** f with **projection** functions

Lemma

$g(x_1, \dots, x_m) = f(y_1, \dots, y_n) \in \text{PR}$ if $f: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $y_i \in \{x_1, \dots, x_m\}$ for all $1 \leq i \leq n$

Proof

- ▶ $y_i = x_j \implies y_i = \pi_j^m(x_1, \dots, x_m)$
- ▶ hence g can be defined by composing f with projection functions

Example

- ▶ **cut-off subtraction** (monus)

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive

Lemma

$g(x_1, \dots, x_m) = f(y_1, \dots, y_n) \in \text{PR}$ if $f: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $y_i \in \{x_1, \dots, x_m\}$ for all $1 \leq i \leq n$

Proof

- ▶ $y_i = x_j \implies y_i = \pi_j^m(x_1, \dots, x_m)$
- ▶ hence g can be defined by composing f with projection functions

Example

- ▶ **cut-off subtraction** (monus)

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} x \dot{-} 0 &= x \\ x \dot{-} (y + 1) &= p(x \dot{-} y) \end{aligned}$$

is primitive recursive

Lemma

$g(x_1, \dots, x_m) = f(y_1, \dots, y_n) \in \text{PR}$ if $f: \mathbb{N}^n \rightarrow \mathbb{N} \in \text{PR}$ and $y_i \in \{x_1, \dots, x_m\}$ for all $1 \leq i \leq n$

Proof

- ▶ $y_i = x_j \implies y_i = \pi_j^m(x_1, \dots, x_m)$
- ▶ hence g can be defined by composing f with projection functions

Example

- ▶ cut-off subtraction (monus)

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} x \dot{-} 0 &= x \\ x \dot{-} (y + 1) &= \text{p}(x \dot{-} y) \end{aligned}$$

is primitive recursive

Examples

► predecessor $p(x) = f(x) \in \text{PR}$

$$f(0) = 0$$

$$f(x+1) = x = \pi_2^2(f(x), x)$$

Examples

- ▶ predecessor $p(x) = f(x) \in \text{PR}$

$$f(0) = 0$$

$$f(x+1) = x = \pi_2^2(f(x), x)$$

- ▶ **factorial** $x! = f(x) \in \text{PR}$

$$f(0) = 1$$

$$f(x+1) = s(x) \times f(x)$$

Examples

- ▶ predecessor $p(x) = f(x) \in \text{PR}$

$$f(0) = 0$$

$$f(x+1) = x = \pi_2^2(f(x), x)$$

- ▶ factorial $x! = f(x) \in \text{PR}$

$$f(0) = 1$$

$$f(x+1) = s(x) \times f(x)$$

- ▶ summation $\sum_{i=1}^x i = f(x) \in \text{PR}$

$$f(0) = 0$$

$$f(x+1) = s(x) + f(x)$$

Examples

- predecessor $p(x) = f(x) \in \text{PR}$

$$f(0) = 0 ?$$

$$f(x+1) = x = \pi_2^2(f(x), x)$$

- factorial $x! = f(x) \in \text{PR}$

$$f(0) = 1 ?$$

$$f(x+1) = s(x) \times f(x)$$

- summation $\sum_{i=1}^x i = f(x) \in \text{PR}$

$$f(0) = 0 ?$$

$$f(x+1) = s(x) + f(x)$$

Definition

function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained from function $g: \mathbb{N} \rightarrow \mathbb{N}$ by **iteration** if

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

Definition

function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained from function $g: \mathbb{N} \rightarrow \mathbb{N}$ by iteration if

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

Lemma

PR is closed under iteration

Definition

function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained from function $g: \mathbb{N} \rightarrow \mathbb{N}$ by iteration if

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

Lemma

PR is closed under iteration

Proof

► suppose $g: \mathbb{N} \rightarrow \mathbb{N} \in \text{PR}$

Definition

function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained from function $g: \mathbb{N} \rightarrow \mathbb{N}$ by iteration if

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

Lemma

PR is closed under iteration

Proof

- ▶ suppose $g: \mathbb{N} \rightarrow \mathbb{N} \in \text{PR}$
- ▶ $f(n, x) = g^{(n)}(x)$ can be defined by primitive recursion:

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

Definition

function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained from function $g: \mathbb{N} \rightarrow \mathbb{N}$ by iteration if

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

Lemma

PR is closed under iteration

Proof

- ▶ suppose $g: \mathbb{N} \rightarrow \mathbb{N} \in \text{PR}$
- ▶ $f(n, x) = g^{(n)}(x)$ can be defined by primitive recursion:

$$\begin{aligned} f(0, x) &= x = \pi_1^1(x) \\ f(n+1, x) &= g(f(n, x)) \end{aligned}$$

Definition

function $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ is obtained from function $g: \mathbb{N} \rightarrow \mathbb{N}$ by iteration if

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

Lemma

PR is closed under iteration

Proof

- ▶ suppose $g: \mathbb{N} \rightarrow \mathbb{N} \in \text{PR}$
- ▶ $f(n, x) = g^{(n)}(x)$ can be defined by primitive recursion:

$$\begin{aligned} f(0, x) &= x = \pi_1^1(x) \\ f(n+1, x) &= g(f(n, x)) = h(f(n, x), n, x) \end{aligned}$$

with $h(x, y, z) = g(\pi_1^3(x, y, z))$

Outline

1. Organisation
2. Contents
3. Primitive Recursive Functions
- 4. Primitive Recursive Predicates**
5. Pairing
6. Summary

Example

► $\max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases}$ is primitive recursive

Example

► $\max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases}$ is primitive recursive ?

Example

► $\max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases}$ is primitive recursive ?

Definition

predicate $P: \mathbb{N}^n \rightarrow \mathbb{B}$ is primitive recursive if its **characteristic function** $\chi_P: \mathbb{N}^n \rightarrow \mathbb{N}$

$$\chi_P(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } P(x_1, \dots, x_n) \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive

Example

► $\max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases}$ is primitive recursive ?

Definition

predicate $P: \mathbb{N}^n \rightarrow \mathbb{B}$ is primitive recursive if its characteristic function $\chi_P: \mathbb{N}^n \rightarrow \mathbb{N}$

$$\chi_P(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } P(x_1, \dots, x_n) \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive

Lemma

if $P, Q: \mathbb{N}^n \rightarrow \mathbb{B}$ are primitive recursive predicates then so are

$\neg P$

$P \wedge Q$

$P \vee Q$

$P \Rightarrow Q$

Proof

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

Proof

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

$$\chi_{P \wedge Q}(\vec{x}) = \chi_P(\vec{x}) \times \chi_Q(\vec{x})$$

Proof

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

$$\chi_{P \wedge Q}(\vec{x}) = \chi_P(\vec{x}) \times \chi_Q(\vec{x})$$

Examples

► **sign** function $\text{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$ is primitive recursive

Proof

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

$$\chi_{P \wedge Q}(\vec{x}) = \chi_P(\vec{x}) \times \chi_Q(\vec{x})$$

Examples

► **sign** function $\text{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$ is primitive recursive

$$\text{sg}(0) = 0$$

$$\text{sg}(x + 1) = 1$$

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

$$\chi_{P \wedge Q}(\vec{x}) = \chi_P(\vec{x}) \times \chi_Q(\vec{x})$$

Examples

- ▶ sign function $\text{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$ is primitive recursive $\begin{array}{l} \text{sg}(0) = 0 \\ \text{sg}(x+1) = 1 \end{array}$
- ▶ $>$ and $\neq$ are primitive recursive predicates

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

$$\chi_{P \wedge Q}(\vec{x}) = \chi_P(\vec{x}) \times \chi_Q(\vec{x})$$

Examples

► sign function $\text{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$ is primitive recursive $\text{sg}(0) = 0$
 $\text{sg}(x + 1) = 1$

► $>$ and $\neq$ are primitive recursive predicates

$$\chi_{>}(x, y) = \text{sg}(x \dot{-} y)$$

$$\chi_{\neq}(x, y) = \text{sg}((x \dot{-} y) + (y \dot{-} x))$$

$$\chi_{\neg P}(\vec{x}) = 1 \dot{-} \chi_P(\vec{x})$$

$$\chi_{P \wedge Q}(\vec{x}) = \chi_P(\vec{x}) \times \chi_Q(\vec{x})$$

Examples

► sign function $\text{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$ is primitive recursive $\text{sg}(0) = 0$
 $\text{sg}(x+1) = 1$

► $>$ and $\neq$ are primitive recursive predicates

$$\chi_{>}(x, y) = \text{sg}(x \dot{-} y)$$

$$\chi_{\neq}(x, y) = \text{sg}((x \dot{-} y) + (y \dot{-} x))$$

► $=$, $\geq$, $<$ and $\leq$ are primitive recursive predicates

$$x = y \iff \neg(x \neq y)$$

$$x < y \iff y > x$$

$$x \geq y \iff x > y \vee x = y$$

$$x \leq y \iff y \geq x$$

Lemma (case analysis)

if $f_1, \dots, f_k: \mathbb{N}^n \rightarrow \mathbb{N}$ and $P_1, \dots, P_k: \mathbb{N}^n \rightarrow \mathbb{B}$ are primitive recursive such that for all $\vec{x} \in \mathbb{N}^n$ exactly one of $P_1(\vec{x}), \dots, P_k(\vec{x})$ holds then

$$g(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{if } P_1(\vec{x}) \\ \dots & \dots \\ f_k(\vec{x}) & \text{if } P_k(\vec{x}) \end{cases}$$

is primitive recursive

Lemma (case analysis)

if $f_1, \dots, f_k: \mathbb{N}^n \rightarrow \mathbb{N}$ and $P_1, \dots, P_k: \mathbb{N}^n \rightarrow \mathbb{B}$ are primitive recursive such that for all $\vec{x} \in \mathbb{N}^n$ exactly one of $P_1(\vec{x}), \dots, P_k(\vec{x})$ holds then

$$g(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{if } P_1(\vec{x}) \\ \dots & \dots \\ f_k(\vec{x}) & \text{if } P_k(\vec{x}) \end{cases}$$

is primitive recursive

Proof

$$g(\vec{x}) = f_1(\vec{x}) \times \chi_{P_1}(\vec{x}) + \dots + f_k(\vec{x}) \times \chi_{P_k}(\vec{x})$$

Lemma (case analysis)

if $f_1, \dots, f_k: \mathbb{N}^n \rightarrow \mathbb{N}$ and $P_1, \dots, P_k: \mathbb{N}^n \rightarrow \mathbb{B}$ are primitive recursive such that for all $\vec{x} \in \mathbb{N}^n$ exactly one of $P_1(\vec{x}), \dots, P_k(\vec{x})$ holds then

$$g(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{if } P_1(\vec{x}) \\ \dots & \dots \\ f_k(\vec{x}) & \text{if } P_k(\vec{x}) \end{cases}$$

is primitive recursive

Proof

$$g(\vec{x}) = f_1(\vec{x}) \times \chi_{P_1}(\vec{x}) + \dots + f_k(\vec{x}) \times \chi_{P_k}(\vec{x})$$

Example

► $\max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{if } x < y \end{cases}$ is primitive recursive

Example

how to implement

$$\text{score} = \min \left(\max \left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B \right), 100 \right)$$

in OLAT ?

Example

how to implement

$$\text{score} = \min \left(\max \left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B \right), 100 \right)$$

in OLAT ?

► `E = getScore("114479890254216")`

Example

how to implement

$$\text{score} = \min \left(\max \left(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B \right), 100 \right)$$

in OLAT ?

- ▶ `E = getScore("114479890254216")`
- ▶ `P = getScore("114479890251762")`
- ▶ `T = getScore("114479890268734")`
- ▶ `B = getScore("114479890265058")`

Example

how to implement

$$\text{score} = \min(\max(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B), 100)$$

in OLAT ?

- ▶ `E = getScore("114479890254216")`
- ▶ `P = getScore("114479890251762")`
- ▶ `T = getScore("114479890268734")`
- ▶ `B = getScore("114479890265058")`

Example

how to implement

$$\text{score} = \min(\max(\frac{2}{3}(E + P) + \frac{1}{3}T + B, T + B), 100)$$

in OLAT ?

- ▶ `E = getScore("114479890254216")`
- ▶ `P = getScore("114479890251762")`
- ▶ `T = getScore("114479890268734")`
- ▶ `B = getScore("114479890265058")`

$$\begin{aligned} &(((2 * (E + P) / 3 + T / 3 + B))) \geq (T + B) * (((2 * (E + P) / 3 + T / 3 + B))) + \\ &(((2 * (E + P) / 3 + T / 3 + B))) < (T + B) * (T + B) \end{aligned}$$

Lemma (bounded sum)

if $f: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ is primitive recursive then

$$\sum_{i=0}^x f(i, \vec{y})$$

is primitive recursive

Lemma (bounded sum)

if $f: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ is primitive recursive then

$$\sum_{i=0}^x f(i, \vec{y})$$

is primitive recursive

Proof

$$g(x, \vec{y}) = \sum_{i=0}^x f(i, \vec{y})$$

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

Lemma (bounded sum and product)

if $f: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ is primitive recursive then

$$\sum_{i=0}^x f(i, \vec{y}) \quad \text{and} \quad \prod_{i=0}^x f(i, \vec{y})$$

are primitive recursive

Proof

$$g(x, \vec{y}) = \sum_{i=0}^x f(i, \vec{y})$$

$$h(x, \vec{y}) = \prod_{i=0}^x f(i, \vec{y})$$

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

$$g(x+1, \vec{y}) = f(x+1, \vec{y}) + g(x, \vec{y})$$

$$h(0, \vec{y}) = f(0, \vec{y})$$

$$h(x+1, \vec{y}) = f(x+1, \vec{y}) \times h(x, \vec{y})$$

Lemma (bounded quantification)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\forall i \leq x) P(i, \vec{y}) \quad \text{and} \quad (\exists i \leq x) P(i, \vec{y})$$

are primitive recursive

Lemma (bounded quantification)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\forall i \leq x) P(i, \vec{y}) \quad \text{and} \quad (\exists i \leq x) P(i, \vec{y})$$

are primitive recursive

Proof

$$Q(x, \vec{y}) = (\forall i \leq x) P(i, \vec{y})$$

$$\chi_Q(x, \vec{y}) = \prod_{i \leq x} \chi_P(i, \vec{y})$$

Lemma (bounded quantification)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\forall i \leq x) P(i, \vec{y}) \quad \text{and} \quad (\exists i \leq x) P(i, \vec{y})$$

are primitive recursive

Proof

$$Q(x, \vec{y}) = (\forall i \leq x) P(i, \vec{y})$$

$$\chi_Q(x, \vec{y}) = \prod_{i \leq x} \chi_P(i, \vec{y})$$

$$R(x, \vec{y}) = (\exists i \leq x) P(i, \vec{y})$$

$$\chi_R(x, \vec{y}) = \text{sg} \left(\sum_{i \leq x} \chi_P(i, \vec{y}) \right)$$

Examples

► x is **divisor** of y

$$x \mid y \iff (\exists i \leq y) [i \times x = y]$$

Examples

- ▶ x is divisor of y $x \mid y \iff (\exists i \leq y) [i \times x = y]$
- ▶ x is **prime number** $\text{prime}(x) \iff x > 1 \wedge (\forall i \leq x) [i \mid x \implies i = 1 \vee i = x]$

Examples

- ▶ x is divisor of y $x \mid y \iff (\exists i \leq y) [i \times x = y]$
- ▶ x is prime number $\text{prime}(x) \iff x > 1 \wedge (\forall i \leq x) [i \mid x \implies i = 1 \vee i = x]$

Lemma (bounded minimization)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\}$$

is primitive recursive

Examples

- ▶ x is divisor of y $x \mid y \iff (\exists i \leq y) [i \times x = y]$
- ▶ x is prime number $\text{prime}(x) \iff x > 1 \wedge (\forall i \leq x) [i \mid x \implies i = 1 \vee i = x]$

Lemma (bounded minimization)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

is primitive recursive

Examples

- ▶ x is divisor of y $x \mid y \iff (\exists i \leq y) [i \times x = y]$
- ▶ x is prime number $\text{prime}(x) \iff x > 1 \wedge (\forall i \leq x) [i \mid x \implies i = 1 \vee i = x]$

Lemma (bounded minimization)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

is primitive recursive

Example

- ▶ $\lfloor \frac{x}{2} \rfloor$ is primitive recursive:

Examples

- ▶ x is divisor of y $x \mid y \iff (\exists i \leq y) [i \times x = y]$
- ▶ x is prime number $\text{prime}(x) \iff x > 1 \wedge (\forall i \leq x) [i \mid x \implies i = 1 \vee i = x]$

Lemma (bounded minimization)

if $P: \mathbb{N}^{n+1} \rightarrow \mathbb{B}$ is primitive recursive then

$$(\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

is primitive recursive

Example

- ▶ $\lfloor \frac{x}{2} \rfloor$ is primitive recursive:

$$\lfloor \frac{x}{2} \rfloor = (\mu i \leq x) [(i + 1) \times 2 > x]$$

Proof

$$f(x, \vec{y}) = (\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

$$f(0, \vec{y}) = \begin{cases} 0 & \text{if } P(0, \vec{y}) \\ 1 & \text{otherwise} \end{cases}$$

$$f(x, \vec{y}) = (\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

$$f(0, \vec{y}) = \begin{cases} 0 & \text{if } P(0, \vec{y}) \\ 1 & \text{otherwise} \end{cases} \quad f(x + 1, \vec{y}) = \begin{cases} f(x, \vec{y}) & \text{if } (\exists i \leq x) P(i, \vec{y}) \\ x + 1 & \text{if } \neg(\exists i \leq x) P(i, \vec{y}) \text{ and } P(x + 1, \vec{y}) \\ x + 2 & \text{otherwise} \end{cases}$$

Proof

$$f(x, \vec{y}) = (\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

$$f(0, \vec{y}) = \begin{cases} 0 & \text{if } P(0, \vec{y}) \\ 1 & \text{otherwise} \end{cases} \quad f(x + 1, \vec{y}) = \begin{cases} f(x, \vec{y}) & \text{if } (\exists i \leq x) P(i, \vec{y}) \\ x + 1 & \text{if } \neg(\exists i \leq x) P(i, \vec{y}) \text{ and } P(x + 1, \vec{y}) \\ x + 2 & \text{otherwise} \end{cases}$$

Examples

► division

$$x \div y = (\mu i \leq x) [(i + 1) \times y > x]$$

Proof

$$f(x, \vec{y}) = (\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

$$f(0, \vec{y}) = \begin{cases} 0 & \text{if } P(0, \vec{y}) \\ 1 & \text{otherwise} \end{cases} \quad f(x + 1, \vec{y}) = \begin{cases} f(x, \vec{y}) & \text{if } (\exists i \leq x) P(i, \vec{y}) \\ x + 1 & \text{if } \neg(\exists i \leq x) P(i, \vec{y}) \text{ and } P(x + 1, \vec{y}) \\ x + 2 & \text{otherwise} \end{cases}$$

Examples

► division

$$x \div y = (\mu i \leq x) [(i + 1) \times y > x]$$

► exponent

$$\exp(x, y) = (\mu i \leq x) [y^i \mid x \wedge \neg(y^{i+1} \mid x)]$$

Proof

$$f(x, \vec{y}) = (\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

$$f(0, \vec{y}) = \begin{cases} 0 & \text{if } P(0, \vec{y}) \\ 1 & \text{otherwise} \end{cases} \quad f(x + 1, \vec{y}) = \begin{cases} f(x, \vec{y}) & \text{if } (\exists i \leq x) P(i, \vec{y}) \\ x + 1 & \text{if } \neg(\exists i \leq x) P(i, \vec{y}) \text{ and } P(x + 1, \vec{y}) \\ x + 2 & \text{otherwise} \end{cases}$$

Examples

- ▶ division $x \div y = (\mu i \leq x) [(i + 1) \times y > x]$
- ▶ exponent $\exp(x, y) = (\mu i \leq x) [y^i \mid x \wedge \neg(y^{i+1} \mid x)]$
- ▶ remainder $x \bmod y = x \dot{-} (y \times (x \div y))$

Proof

$$f(x, \vec{y}) = (\mu i \leq x) P(i, \vec{y}) = \min \{i \mid 0 \leq i \leq x \wedge P(i, \vec{y})\} \cup \{x + 1\}$$

$$f(0, \vec{y}) = \begin{cases} 0 & \text{if } P(0, \vec{y}) \\ 1 & \text{otherwise} \end{cases} \quad f(x + 1, \vec{y}) = \begin{cases} f(x, \vec{y}) & \text{if } (\exists i \leq x) P(i, \vec{y}) \\ x + 1 & \text{if } \neg(\exists i \leq x) P(i, \vec{y}) \text{ and } P(x + 1, \vec{y}) \\ x + 2 & \text{otherwise} \end{cases}$$

Examples

- ▶ division $x \div y = (\mu i \leq x) [(i + 1) \times y > x]$
- ▶ exponent $\exp(x, y) = (\mu i \leq x) [y^i \mid x \wedge \neg(y^{i+1} \mid x)]$
- ▶ remainder $x \bmod y = x \dot{-} (y \times (x \div y))$
- ▶ n -th **prime number** p_n $p_0 = 2$ with $f(x) = g(x! + 1, x)$
 $p_{n+1} = f(p_n)$ $g(x, y) = (\mu i \leq x) [\text{prime}(i) \wedge i > y]$

Remark

replacing $i \leq x$ by $i < x$ does not affect closure under bounded minimization

Proof

$$g(x, \vec{y}) = \sum_{i < x} f(i, \vec{y})$$

Remark

replacing $i \leq x$ by $i < x$ does not affect closure under bounded minimization

Proof

$$g(x, \vec{y}) = \sum_{i < x} f(i, \vec{y})$$

$$g(0, \vec{y}) = 0$$

$$g(x+1, \vec{y}) = f(x, \vec{y}) + g(x, \vec{y})$$

Remark

replacing $i \leq x$ by $i < x$ does not affect closure under bounded minimization

Proof

$$g(x, \vec{y}) = \sum_{i < x} f(i, \vec{y})$$

$$g(0, \vec{y}) = 0$$

$$g(x+1, \vec{y}) = f(x, \vec{y}) + g(x, \vec{y})$$

$$Q(x, \vec{y}) = (\exists i < x) P(i, \vec{y})$$

Remark

replacing $i \leq x$ by $i < x$ does not affect closure under bounded minimization

Proof

$$g(x, \vec{y}) = \sum_{i < x} f(i, \vec{y})$$

$$g(0, \vec{y}) = 0$$

$$g(x+1, \vec{y}) = f(x, \vec{y}) + g(x, \vec{y})$$

$$Q(x, \vec{y}) = (\exists i < x) P(i, \vec{y})$$

$$\chi_Q(x, \vec{y}) = \text{sg} \left(\sum_{i < x} \chi_P(i, \vec{y}) \right)$$

Remark

replacing $i \leq x$ by $i < x$ does not affect closure under bounded minimization

Proof

$$g(x, \vec{y}) = \sum_{i < x} f(i, \vec{y})$$

$$g(0, \vec{y}) = 0$$

$$g(x+1, \vec{y}) = f(x, \vec{y}) + g(x, \vec{y})$$

$$Q(x, \vec{y}) = (\exists i < x) P(i, \vec{y})$$

$$\chi_Q(x, \vec{y}) = \text{sg} \left(\sum_{i < x} \chi_P(i, \vec{y}) \right)$$

$$f(x, \vec{y}) = (\mu i < x) P(i, \vec{y}) = \min \{i \mid 0 \leq i < x \wedge P(i, \vec{y})\} \cup \{x\}$$

Remark

replacing $i \leq x$ by $i < x$ does not affect closure under bounded minimization

Proof

$$g(x, \vec{y}) = \sum_{i < x} f(i, \vec{y})$$

$$g(0, \vec{y}) = 0$$

$$g(x+1, \vec{y}) = f(x, \vec{y}) + g(x, \vec{y})$$

$$Q(x, \vec{y}) = (\exists i < x) P(i, \vec{y})$$

$$\chi_Q(x, \vec{y}) = \text{sg} \left(\sum_{i < x} \chi_P(i, \vec{y}) \right)$$

$$f(x, \vec{y}) = (\mu i < x) P(i, \vec{y}) = \min \{i \mid 0 \leq i < x \wedge P(i, \vec{y})\} \cup \{x\}$$

$$f(0, \vec{y}) = 0$$

$$f(x+1, \vec{y}) = \begin{cases} f(x, \vec{y}) & \text{if } (\exists i < x) P(i, \vec{y}) \\ x & \text{if } \neg(\exists i < x) P(i, \vec{y}) \text{ and } P(x, \vec{y}) \\ x+1 & \text{otherwise} \end{cases}$$

Outline

1. Organisation
2. Contents
3. Primitive Recursive Functions
4. Primitive Recursive Predicates
- 5. Pairing**
6. Summary

Example

- Fibonacci function $\text{fib}(x)$

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

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

$$\text{fib}(x + 2) = \text{fib}(x + 1) + \text{fib}(x)$$

is primitive recursive

Example

- Fibonacci function $\text{fib}(x)$

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

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

$$\text{fib}(x + 2) = \text{fib}(x + 1) + \text{fib}(x) \quad \text{course-of-values recursion}$$

is primitive recursive ?

Example

► Fibonacci function $\text{fib}(x)$

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

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

$$\text{fib}(x + 2) = \text{fib}(x + 1) + \text{fib}(x) \quad \text{course-of-values recursion}$$

is primitive recursive ?

Idea

combine $\text{fib}(x + 1)$ and $\text{fib}(x)$ into single number from which $\text{fib}(x + 1)$ and $\text{fib}(x)$ can be obtained by suitable primitive recursive **extraction** functions

Definitions

► **pairing** function $\pi(x, y) = 2^x(2y + 1) \dot{-} 1$

Definitions

- ▶ pairing function $\pi(x, y) = 2^x(2y + 1) \dot{-} 1$
- ▶ **extraction** functions $\pi_1(z) = (\mu x \leq z) (\exists y \leq z) [z = \pi(x, y)]$
 $\pi_2(z) = (\mu y \leq z) (\exists x \leq z) [z = \pi(x, y)]$

Definitions

- ▶ pairing function $\pi(x, y) = 2^x(2y + 1) \dot{-} 1$
- ▶ extraction functions $\pi_1(z) = (\mu x \leq z) (\exists y \leq z) [z = \pi(x, y)]$
 $\pi_2(z) = (\mu y \leq z) (\exists x \leq z) [z = \pi(x, y)]$

Lemmata

- 1 $x, y \leq \pi(x, y)$

Definitions

- ▶ pairing function $\pi(x, y) = 2^x(2y + 1) \dot{-} 1$
- ▶ extraction functions $\pi_1(z) = (\mu x \leq z) (\exists y \leq z) [z = \pi(x, y)]$
 $\pi_2(z) = (\mu y \leq z) (\exists x \leq z) [z = \pi(x, y)]$

Lemmata

- 1 $x, y \leq \pi(x, y)$
- 2 $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$

Definitions

- ▶ pairing function $\pi(x, y) = 2^x(2y + 1) \dot{-} 1$
- ▶ extraction functions $\pi_1(z) = (\mu x \leq z) (\exists y \leq z) [z = \pi(x, y)]$
 $\pi_2(z) = (\mu y \leq z) (\exists x \leq z) [z = \pi(x, y)]$

Lemmata

- 1 $x, y \leq \pi(x, y)$
- 2 $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$
- 3 $\pi(\pi_1(z), \pi_2(z)) = z$

Definitions

- ▶ pairing function $\pi(x, y) = 2^x(2y + 1) \dot{-} 1$
- ▶ extraction functions $\pi_1(z) = (\mu x \leq z) (\exists y \leq z) [z = \pi(x, y)]$
 $\pi_2(z) = (\mu y \leq z) (\exists x \leq z) [z = \pi(x, y)]$

Lemmata

- 1 $x, y \leq \pi(x, y)$
- 2 $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$
- 3 $\pi(\pi_1(z), \pi_2(z)) = z$
- 4 π, π_1, π_2 are primitive recursive

Lemmata

$$\textcircled{1} \quad x, y \leq \pi(x, y)$$

Proof

$$\textcircled{1} \quad \pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq 2^x \dot{-} 1 \geq x$$

Lemmata

$$\textcircled{1} \quad x, y \leq \pi(x, y)$$

Proof

$$\textcircled{1} \quad \pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq 2^x \dot{-} 1 \geq x$$

$$\pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq (2y + 1) \dot{-} 1 = 2y \geq y$$

Lemmata

- ① $x, y \leq \pi(x, y)$
- ② $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$

Proof

① $\pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq 2^x \dot{-} 1 \geq x$

$$\pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq (2y + 1) \dot{-} 1 = 2y \geq y$$

② $\pi_1(\pi(x, y)) = (\mu x' \leq \pi(x, y)) (\exists y' \leq \pi(x, y)) [\pi(x, y) = \pi(x', y')]]$

Lemmata

- ① $x, y \leq \pi(x, y)$
- ② $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$

Proof

① $\pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq 2^x \dot{-} 1 \geq x$

$$\pi(x, y) = 2^x(2y + 1) \dot{-} 1 \geq (2y + 1) \dot{-} 1 = 2y \geq y$$

② $\pi_1(\pi(x, y)) = (\mu x' \leq \pi(x, y)) (\exists y' \leq \pi(x, y)) [\pi(x, y) = \pi(x', y')]]$

$$\pi \text{ is injective} \implies \pi(x, y) = \pi(x', y') \text{ only holds when } x' = x \text{ and } y' = y$$

Lemmata

- ① $x, y \leq \pi(x, y)$
- ② $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$
- ③ $\pi(\pi_1(z), \pi_2(z)) = z$

Proof

- ① $\pi(x, y) = 2^x(2y + 1) \dot{\div} 1 \geq 2^x \dot{\div} 1 \geq x$
 $\pi(x, y) = 2^x(2y + 1) \dot{\div} 1 \geq (2y + 1) \dot{\div} 1 = 2y \geq y$
- ② $\pi_1(\pi(x, y)) = (\mu x' \leq \pi(x, y)) (\exists y' \leq \pi(x, y)) [\pi(x, y) = \pi(x', y')]$
 π is injective $\implies \pi(x, y) = \pi(x', y')$ only holds when $x' = x$ and $y' = y$
- ③ π is surjective $\implies z = \pi(x, y)$ for some x and y

Lemmata

- ① $x, y \leq \pi(x, y)$
- ② $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$
- ③ $\pi(\pi_1(z), \pi_2(z)) = z$

Proof

- ① $\pi(x, y) = 2^x(2y + 1) \dot{\div} 1 \geq 2^x \dot{\div} 1 \geq x$
 $\pi(x, y) = 2^x(2y + 1) \dot{\div} 1 \geq (2y + 1) \dot{\div} 1 = 2y \geq y$
- ② $\pi_1(\pi(x, y)) = (\mu x' \leq \pi(x, y)) (\exists y' \leq \pi(x, y)) [\pi(x, y) = \pi(x', y')]$
 π is injective $\implies \pi(x, y) = \pi(x', y')$ only holds when $x' = x$ and $y' = y$
- ③ π is surjective $\implies z = \pi(x, y)$ for some x and y
 $\pi_1(z) = x$ and $\pi_2(z) = y$

Lemmata

- ① $x, y \leq \pi(x, y)$
- ② $\pi_1(\pi(x, y)) = x$ and $\pi_2(\pi(x, y)) = y$
- ③ $\pi(\pi_1(z), \pi_2(z)) = z$

Proof

- ① $\pi(x, y) = 2^x(2y + 1) \dot{\div} 1 \geq 2^x \dot{\div} 1 \geq x$
 $\pi(x, y) = 2^x(2y + 1) \dot{\div} 1 \geq (2y + 1) \dot{\div} 1 = 2y \geq y$
- ② $\pi_1(\pi(x, y)) = (\mu x' \leq \pi(x, y)) (\exists y' \leq \pi(x, y)) [\pi(x, y) = \pi(x', y')]$
 π is injective $\implies \pi(x, y) = \pi(x', y')$ only holds when $x' = x$ and $y' = y$
- ③ π is surjective $\implies z = \pi(x, y)$ for some x and y
 $\pi_1(z) = x$ and $\pi_2(z) = y \implies z = \pi(\pi_1(z), \pi_2(z))$

Lemma

fib is primitive recursive

Proof

- ▶ $g(x) = \pi(\text{fib}(x), \text{fib}(x + 1))$ is primitive recursive

Lemma

fib is primitive recursive

Proof

► $g(x) = \pi(\text{fib}(x), \text{fib}(x + 1))$ is primitive recursive:

$$g(0) = \pi(1, 1)$$

$$g(x + 1) = \pi(\pi_2(g(x)), \pi_1(g(x)) + \pi_2(g(x)))$$

Lemma

fib is primitive recursive

Proof

- ▶ $g(x) = \pi(\text{fib}(x), \text{fib}(x + 1))$ is primitive recursive:

$$g(0) = \pi(1, 1)$$

$$g(x + 1) = \pi(\pi_2(g(x)), \pi_1(g(x)) + \pi_2(g(x)))$$

- ▶ $\text{fib}(x) = \pi_1(g(x))$

Outline

1. Organisation
2. Contents
3. Primitive Recursive Functions
4. Primitive Recursive Predicates
5. Pairing
- 6. Summary**

Important Concepts

- ▶ bounded minimization
- ▶ bounded quantification
- ▶ case analysis
- ▶ characteristic function
- ▶ composition
- ▶ initial function
- ▶ iteration
- ▶ pairing
- ▶ PR
- ▶ primitive recursion

Important Concepts

- ▶ bounded minimization
- ▶ bounded quantification
- ▶ case analysis
- ▶ characteristic function
- ▶ composition
- ▶ initial function
- ▶ iteration
- ▶ pairing
- ▶ PR
- ▶ primitive recursion

homework for October 12

Important Concepts

- ▶ bounded minimization
- ▶ bounded quantification
- ▶ case analysis
- ▶ characteristic function
- ▶ composition
- ▶ initial function
- ▶ iteration
- ▶ pairing
- ▶ PR
- ▶ primitive recursion

homework for October 12

deregistration (without grade) possible until October 19