universität innsbruck

WS 2023 lecture 4

Computability Theory

Aart Middeldorp

Outline

1. **Summary of Previous Lecture**
2. **Recursive Functions**
3. **While Programs**
4. **Partial Recursive Functions**
5. **Normal Form Theorem**
6. **Summary**

Definition

function $f\colon \mathbb{N}^n \to \mathbb{N}$ is LOOP computable if $\exists$ LOOP program $P(x_1,\dots,x_n;y)$ such that $y = f(x_1,\dots,x_n)$ after execution of P

Theorem

primitive recursive functions are LOOP computable

Definitions

- class $\mathcal{E}$ of elementary functions is smallest class of (total) functions $f\colon \mathbb{N}^n \to \mathbb{N}$ that contains all initial functions, $+$, $\dot{-}$ and is closed under composition, bounded summation and bounded product
- binary function $2_x(y)$ is defined by primitive recursion

$$2_0(y) = y \qquad\qquad 2_{x+1}(y) = 2^{2_x(y)}$$

Lemma

$\forall$ elementary function $f\colon \mathbb{N}^n \to \mathbb{N}$ $\exists$ constant $c \in \mathbb{N}$ such that

$$f(x_1, \dots, x_n) < 2_c(\max\{x_1, \dots, x_n\})$$

Corollary

$\mathsf{E} \subsetneq \mathsf{PR}$

Definition (bounded recursion)

class C of numeric functions is closed under bounded recursion if $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by primitive recursion from $g\colon \mathbb{N}^n \to \mathbb{N} \in \mathsf{C}$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N} \in \mathsf{C}$ and satisfying

$$f(x, \vec{y}) \leqslant i(x, \vec{y})$$

for some $i\colon \mathbb{N}^{n+1} \to \mathbb{N} \in \mathsf{C}$ different from f, belongs to C

Definitions

- $\mathrm{e}_0(x,y) = x + y \qquad \mathrm{e}_1(x) = x^2 + 2 \qquad \mathrm{e}_{n+2}(x) = \begin{cases} 2 & \text{if } x = 0 \\ \mathrm{e}_{n+1}(\mathrm{e}_{n+2}(x-1)) & \text{if } x > 0 \end{cases}$
- E_0 is smallest class of functions that contains all initial functions and is closed under composition and bounded recursion
- E_{n+1} is smallest class of functions that contains all initial functions, e_0, e_n and is closed under composition and bounded recursion

Theorem (Grzegorczyk Hierarchy)

1. $\mathrm{E}_0 \subsetneq \mathrm{E}_1 \subsetneq \mathrm{E}_2 \subsetneq \cdots$
2. $\mathrm{E}_3 = \mathrm{E}$
3. $\bigcup_{n \geqslant 0} \mathrm{E}_n = \mathrm{PR}$

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 and Lambda Calculus

α−equivalence, abstraction, arithmetization, β−reduction, CL−representability, combinators, combinatorial completeness, Church numerals, Church−Rosser theorem, Curry−Howard isomorphism, de Bruijn notation, η−reduction, fixed point theorem, intuitionistic propositional logic, λ−definability, 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 and Lambda Calculus

α−equivalence, abstraction, arithmetization, β−reduction, CL−representability, combinators, combinatorial completeness, Church numerals, Church−Rosser theorem, Curry−Howard isomorphism, de Bruijn notation, η−reduction, fixed point theorem, intuitionistic propositional logic, λ−definability, normalization theorem, termination, typing, undecidability, Z property, ...

Outline

1. Summary of Previous Lecture
2. **Recursive Functions**
3. While Programs
4. Partial Recursive Functions
5. Normal Form Theorem
6. Summary

Definition

class R of recursive functions is smallest class of total functions that contains all initial functions and is closed under composition, primitive recursion, and minimization:

$$(\mu\, i)\,(f(i,\vec{y}) = 0) \in \mathrm{R}$$

for all $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ in R

Definition

class R of recursive functions is smallest class of total functions that contains all initial functions and is closed under composition, primitive recursion, and minimization:

$$(\mu\, i)\ (f(i,\vec{y}) = 0) \in \mathrm{R}$$

for all $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ in R

Theorem

R is smallest class of total functions that contains

- all projection functions

Definition

class R of recursive functions is smallest class of total functions that contains all initial functions and is closed under composition, primitive recursion, and minimization:

$$(\mu\, i)\,(f(i,\vec{y}) = 0) \in \mathrm{R}$$

for all $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ in R

Theorem

R is smallest class of total functions that contains

- all projection functions
- addition and multiplication

Definition

class R of recursive functions is smallest class of total functions that contains all initial functions and is closed under composition, primitive recursion, and minimization:

$$(\mu\, i)\ (f(i, \vec{y}) = 0) \in \mathsf{R}$$

for all $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ in R

Theorem

R is smallest class of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

Definition

class R of recursive functions is smallest class of total functions that contains all initial functions and is closed under composition, primitive recursion, and minimization:

$$(\mu\, i)\ (f(i, \vec{y}) = 0) \in \mathrm{R}$$

for all $f \colon \mathbb{N}^{n+1} \to \mathbb{N}$ in R

Theorem

R is smallest class of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_{=}$ of equality predicate

and is closed under composition and minimization

Theorem

R is smallest class $\mathcal{C}$ of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

and is closed under composition and minimization

Proof

- $\mathcal{C} \subseteq$ R

Theorem

R is smallest class $\mathcal{C}$ of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

and is closed under composition and minimization

Proof

- $\mathcal{C} \subseteq \mathrm{R}$
- $z(x) = 0 = (\mu\, y)\ (\pi_1^2(y, x) = 0) \in \mathcal{C}$

Theorem

R is smallest class $\mathcal{C}$ of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

and is closed under composition and minimization

Proof

- $\mathcal{C} \subseteq \mathrm{R}$
- $\mathrm{z}(x) = 0 = (\mu\, y)\,(\pi^2_1(y, x) = 0) \in \mathcal{C}$
- $\mathrm{s}(x) = x + 1 = \pi^1_1(x) + \chi_=(x, x)$

Theorem

R is smallest class $\mathcal{C}$ of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

and is closed under composition and minimization

Proof

- $\mathcal{C} \subseteq \mathrm{R}$
- $\mathsf{z}(x) = 0 = (\mu\, y)\, (\pi^2_1(y, x) = 0) \in \mathcal{C}$
- $\mathsf{s}(x) = x + 1 = \pi^1_1(x) + \chi_=(x, x)$

Theorem

R is smallest class $\mathcal{C}$ of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

and is closed under composition and minimization

Proof

- $\mathcal{C} \subseteq$ R
- $z(x) = 0 = (\mu\, y)\, (\pi^2_1(y, x) = 0) \in \mathcal{C}$
- $s(x) = x + 1 = \pi^1_1(x) + \chi_=(x, x) = \pi^1_1(x) + \chi_=(\pi^1_1(x), \pi^1_1(x)) \in \mathcal{C}$

Theorem

R is smallest class $\mathcal{C}$ of total functions that contains

- all projection functions
- addition and multiplication
- characteristic function $\chi_=$ of equality predicate

and is closed under composition and minimization

Proof

- $\mathcal{C} \subseteq \mathrm{R}$
- $\mathsf{z}(x) = 0 = (\mu\, y)\, (\pi^2_1(y, x) = 0) \in \mathcal{C}$
- $\mathsf{s}(x) = x + 1 = \pi^1_1(x) + \chi_=(x, x) = \pi^1_1(x) + \chi_=(\pi^1_1(x), \pi^1_1(x)) \in \mathcal{C}$
- $\mathcal{C}$ is closed under primitive recursion ...

Definition

$\mathcal{C}_P$ is class of predicates whose characteristic function belongs to $\mathcal{C}$

Definition

$\mathcal{C}_P$ is class of predicates whose characteristic function belongs to $\mathcal{C}$

Lemma

$\mathcal{C}_P$ is closed under boolean operations

Definition

$\mathcal{C}_P$ is class of predicates whose characteristic function belongs to $\mathcal{C}$

Lemma

$\mathcal{C}_P$ is closed under boolean operations

Proof

$\mathcal{C}_P$ is closed under negation

$$\chi_{\neg P}(x_1, \ldots, x_n) = \chi_{=}(\chi_P(x_1, \ldots, x_n), 0)$$

Definition

$\mathcal{C}_P$ is class of predicates whose characteristic function belongs to $\mathcal{C}$

Lemma

$\mathcal{C}_P$ is closed under boolean operations

Proof

$\mathcal{C}_P$ is closed under negation

$$\chi_{\neg P}(x_1, \dots, x_n) = \chi_{=}(\chi_P(x_1, \dots, x_n), 0) = \chi_{=}(\chi_P(x_1, \dots, x_n), z(\pi_1^n(x_1, \dots, x_n)))$$

Definition

$\mathcal{C}_P$ is class of predicates whose characteristic function belongs to $\mathcal{C}$

Lemma

$\mathcal{C}_P$ is closed under boolean operations

Proof

$\mathcal{C}_P$ is closed under negation

$$\chi_{\neg P}(x_1, \dots, x_n) = \chi_{=}(\chi_P(x_1, \dots, x_n), 0) = \chi_{=}(\chi_P(x_1, \dots, x_n), z(\pi_1^n(x_1, \dots, x_n)))$$

and disjunction

$$\chi_{P \vee Q}(x_1, \dots, x_n) = \chi_{=}(\chi_{=}(\chi_P(x_1, \dots, x_n) + \chi_Q(x_1, \dots, x_n), 0), 0)$$

Definition

$\mathcal{C}_P$ is class of predicates whose characteristic function belongs to $\mathcal{C}$

Lemma

$\mathcal{C}_P$ is closed under boolean operations

Proof

$\mathcal{C}_P$ is closed under negation

$$\chi_{\neg P}(x_1,\dots,x_n) = \chi_{=}(\chi_P(x_1,\dots,x_n),0) = \chi_{=}(\chi_P(x_1,\dots,x_n),z(\pi_1^n(x_1,\dots,x_n)))$$

and disjunction

$$\chi_{P\vee Q}(x_1,\dots,x_n) = \chi_{=}(\chi_{=}(\chi_P(x_1,\dots,x_n)+\chi_Q(x_1,\dots,x_n),0),0)$$

and hence under conjunction

$$\chi_{P\wedge Q}(x_1,\dots,x_n) = \chi_{\neg(\neg P\vee\neg Q)}(x_1,\dots,x_n)$$

Lemma

$\mathcal{C}_P$ is closed under bounded universal and existential quantification

Lemma

$\mathcal{C}_P$ is closed under bounded universal and existential quantification

Proof

- bounded universal quantification

$$Q(x, \vec{y}) = (\forall\, i \leqslant x)\, P(i, \vec{y}) \quad \text{with } P \in \mathcal{C}_P$$

Lemma

$\mathcal{C}_P$ is closed under bounded universal and existential quantification

Proof

- bounded universal quantification

$$Q(x, \vec{y}) = (\forall\, i \leqslant x)\, P(i, \vec{y}) \quad \text{with } P \in \mathcal{C}_P$$

$$\chi_Q(x, \vec{y}) = \chi_=((\mu\, i)\, (\chi_P(i, \vec{y}) = 0 \vee i = x + 1), x + 1)$$

Lemma

$\mathcal{C}_P$ is closed under bounded universal and existential quantification

Proof

- bounded universal quantification

$$Q(x,\vec{y}) = (\forall\, i \leqslant x)\; P(i,\vec{y}) \quad \text{with } P \in \mathcal{C}_P$$

$$\chi_Q(x,\vec{y}) = \chi_=((\mu\, i)\; (\chi_P(i,\vec{y}) = 0 \;\vee\; i = x+1),\; x+1) \in \mathcal{C}$$

Lemma

$\mathcal{C}_P$ is closed under bounded universal and existential quantification

Proof

- bounded universal quantification

$$Q(x,\vec{y}) = (\forall\, i \leqslant x)\, P(i,\vec{y}) \quad \text{with } P \in \mathcal{C}_P$$

$$\chi_Q(x,\vec{y}) = \chi_=((\mu\, i)\, (\chi_P(i,\vec{y}) = 0 \lor i = x+1), x+1) \in \mathcal{C}$$

- bounded existential quantification

$$R(x,\vec{y}) = (\exists\, i \leqslant x)\, P(i,\vec{y}) \quad \text{with } P \in \mathcal{C}_P$$

Lemma

$\mathcal{C}_P$ is closed under bounded universal and existential quantification

Proof

- bounded universal quantification

$$Q(x,\vec{y}) = (\forall\, i \leqslant x)\; P(i,\vec{y}) \quad \text{with } P \in \mathcal{C}_P$$

$$\chi_Q(x,\vec{y}) = \chi_=((\mu\, i)\; (\chi_P(i,\vec{y}) = 0 \;\vee\; i = x+1),\, x+1) \in \mathcal{C}$$

- bounded existential quantification

$$\begin{aligned} R(x,\vec{y}) &= (\exists\, i \leqslant x)\; P(i,\vec{y}) \quad \text{with } P \in \mathcal{C}_P \\ &= \neg\, (\forall\, i \leqslant x)\; \neg P(i,\vec{y}) \end{aligned}$$

Definition

$\beta(a,i) = \pi_1(a) \bmod \left(1 + (i+1)\,\pi_2(a)\right)$ Gödel's β function

Definition

$\beta(a,i) = \pi_1(a) \bmod \big(1 + (i+1)\,\pi_2(a)\big)$ Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Definition

$\beta(a, i) = \pi_1(a) \bmod \big(1 + (i+1)\,\pi_2(a)\big)$ Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Proof

- $x \div 2 = (\mu\, y)\,(2y = x \vee 2y + 1 = x)$

Definition

$\beta(a,i) = \pi_1(a) \bmod \left(1 + (i+1)\,\pi_2(a)\right)$ Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Proof

- $x \div 2 = (\mu\, y)\,(2y = x \vee 2y + 1 = x) \in \mathcal{C}$

Definition

$\beta(a,i) = \pi_1(a) \bmod \big(1 + (i+1)\,\pi_2(a)\big)$ Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Proof

- $x \div 2 = (\mu\, y)\,(2y = x \vee 2y + 1 = x) \in \mathcal{C}$
- $\pi(x,y) = ((x+y)^2 + 3x + y) \div 2$ Cantor pairing function

Definition

$\beta(a,i) = \pi_1(a) \bmod \big(1 + (i+1)\,\pi_2(a)\big)$ Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Proof

- $x \div 2 = (\mu\, y)\,(2y = x \lor 2y + 1 = x) \in \mathcal{C}$
- $\pi(x,y) = ((x+y)^2 + 3x + y) \div 2 \in \mathcal{C}$ Cantor pairing function

Definition

$\beta(a,i) = \pi_1(a) \bmod \big(1 + (i+1)\,\pi_2(a)\big)$ Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Proof

- $x \div 2 = (\mu\, y)\,(2y = x \vee 2y + 1 = x) \in \mathcal{C}$
- $\pi(x,y) = ((x+y)^2 + 3x + y) \div 2 \in \mathcal{C}$ Cantor pairing function
- $\pi_1(a) = (\mu\, x \leqslant a)\,(\exists\, y \leqslant a)\,\big[\, a = \pi(x,y) \,\big] \in \mathcal{C}$
- $\pi_2(a) = (\mu\, y \leqslant a)\,(\exists\, x \leqslant a)\,\big[\, a = \pi(x,y) \,\big] \in \mathcal{C}$

Definition

$\beta(a,i) = \pi_1(a) \bmod \big(1 + (i+1)\,\pi_2(a)\big)$ — Gödel's β function

Lemma

$\beta \in \mathcal{C}$

Proof

- $x \div 2 = (\mu\, y)\,(2y = x \vee 2y + 1 = x) \in \mathcal{C}$
- $\pi(x,y) = ((x+y)^2 + 3x + y) \div 2 \in \mathcal{C}$ — Cantor pairing function
- $\pi_1(a) = (\mu\, x \leqslant a)\,(\exists\, y \leqslant a)\,\big[\, a = \pi(x,y)\,\big] \in \mathcal{C}$
- $\pi_2(a) = (\mu\, y \leqslant a)\,(\exists\, x \leqslant a)\,\big[\, a = \pi(x,y)\,\big] \in \mathcal{C}$
- $x \bmod y = (\mu\, z < y)\,(\exists\, q \leqslant x)\,\big[\, x = qy + z\,\big] \in \mathcal{C}$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

with $g, h \in \mathcal{C}$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\,\big[\,\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x)\,\big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)\,\big]$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\,\big[\,\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x)\,\big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)\,\big] \in \mathcal{C}$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\, \big[\, \beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x)\, \big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)\, \big] \in \mathcal{C}$
- $z = \hat{f}(x',\vec{y})$ satisfies $\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x')\, \big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)$

Proof (cont'd)

- primitive recursion

$$f(0, \vec{y}) = g(\vec{y}) \qquad\qquad f(x+1, \vec{y}) = h(f(x, \vec{y}), x, \vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x, \vec{y}) = (\mu\, z)\ \big[\ \beta(z, 0) = g(\vec{y}) \wedge (\forall\, i < x)\ \big(\beta(z, i+1) = h(\beta(z, i), i, \vec{y})\big)\ \big] \in \mathcal{C}$

- $z = \hat{f}(x', \vec{y})$ satisfies $\beta(z, 0) = g(\vec{y}) \wedge (\forall\, i < x')\ \big(\beta(z, i+1) = h(\beta(z, i), i, \vec{y})\big)$

- claim: $f(x, \vec{y}) = \beta(\hat{f}(x', \vec{y}), x) \ \ \forall\, x' \geqslant x$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\,\big[\,\beta(z,0) = g(\vec{y}) \land (\forall\, i < x)\,\big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)\,\big] \in \mathcal{C}$
- $z = \hat{f}(x',\vec{y})$ satisfies $\beta(z,0) = g(\vec{y}) \land (\forall\, i < x')\,\big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)$
- claim: $f(x,\vec{y}) = \beta(\hat{f}(x',\vec{y}),x) \;\; \forall\, x' \geqslant x$, by induction on x
  - $f(0,\vec{y}) = g(\vec{y}) = \beta(\hat{f}(x',\vec{y}),0) \quad \forall\, x' \geqslant 0$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\left[\,\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x)\left(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\right)\,\right] \in \mathcal{C}$
- $z = \hat{f}(x',\vec{y})$ satisfies $\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x')\left(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\right)$
- claim: $f(x,\vec{y}) = \beta(\hat{f}(x',\vec{y}),x) \;\; \forall\, x' \geqslant x$, by induction on x
  - $f(0,\vec{y}) = g(\vec{y}) = \beta(\hat{f}(x',\vec{y}),0) \quad \forall\, x' \geqslant 0$
  - $f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y}) = h(\beta(\hat{f}(x',\vec{y}),x),x,\vec{y}) \qquad \forall\, x' \geqslant x+1$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\,\big[\, \beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x)\,\big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)\,\big] \in \mathcal{C}$
- $z = \hat{f}(x',\vec{y})$ satisfies $\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x')\,\big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)$
- claim: $f(x,\vec{y}) = \beta(\hat{f}(x',\vec{y}),x) \;\; \forall\, x' \geqslant x$, by induction on x
  - $f(0,\vec{y}) = g(\vec{y}) = \beta(\hat{f}(x',\vec{y}),0) \quad \forall\, x' \geqslant 0$
  - $f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y}) = h(\beta(\hat{f}(x',\vec{y}),x),x,\vec{y}) = \beta(\hat{f}(x',\vec{y}),x+1) \quad \forall\, x' \geqslant x+1$

Proof (cont'd)

- primitive recursion

$$f(0,\vec{y}) = g(\vec{y}) \qquad\qquad f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y})$$

  with $g, h \in \mathcal{C}$

- $\hat{f}(x,\vec{y}) = (\mu\, z)\, \big[\, \beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x)\, \big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)\, \big] \in \mathcal{C}$
- $z = \hat{f}(x',\vec{y})$ satisfies $\beta(z,0) = g(\vec{y}) \wedge (\forall\, i < x')\, \big(\beta(z,i+1) = h(\beta(z,i),i,\vec{y})\big)$
- claim: $f(x,\vec{y}) = \beta(\hat{f}(x',\vec{y}),x)\ \ \forall\, x' \geqslant x$, by induction on x
  - $f(0,\vec{y}) = g(\vec{y}) = \beta(\hat{f}(x',\vec{y}),0) \quad \forall\, x' \geqslant 0$
  - $f(x+1,\vec{y}) = h(f(x,\vec{y}),x,\vec{y}) = h(\beta(\hat{f}(x',\vec{y}),x),x,\vec{y}) = \beta(\hat{f}(x',\vec{y}),x+1) \quad \forall\, x' \geqslant x+1$
- $f(x,\vec{y}) = \beta(\hat{f}(x,\vec{y}),x) \in \mathcal{C}$

Outline

1. **Summary of Previous Lecture**

2. **Recursive Functions**

3. **While Programs**

4. **Partial Recursive Functions**

5. **Normal Form Theorem**

6. **Summary**

Loop Programs

- natural numbers are only data type
- variables x, y, z, $\ldots$
- commands
  - assignment $x := 0$ $x := y$
  - increment $x{+}{+}$
  - composition $P\,;Q$
  - loops
    - LOOP x DO P OD

    execute P exactly n times, where n is value of x before entering loop

While Programs

- natural numbers are only data type
- variables x, y, z, $\ldots$
- commands
  - assignment $x := 0$ $x := y$
  - increment $x{+}{+}$
  - composition $P\,;Q$
  - loops
    - LOOP x DO P OD

      execute P exactly n times, where n is value of x before entering loop
    - WHILE $x > 0$ DO P OD

      repeatedly execute P while $x > 0$

Definition

function $f \colon \mathbb{N}^n \to \mathbb{N}$ is WHILE computable if $\exists$ WHILE program $P(x_1, \ldots, x_n; y)$ such that $y = f(x_1, \ldots, x_n)$ after execution of P

Definition

function $f\colon \mathbb{N}^n \to \mathbb{N}$ is WHILE computable if $\exists$ WHILE program $P(x_1, \ldots, x_n; y)$ such that $y = f(x_1, \ldots, x_n)$ after execution of P

Theorem

recursive functions are WHILE computable

Definition

function $f\colon \mathbb{N}^n \to \mathbb{N}$ is WHILE computable if $\exists$ WHILE program $P(x_1, \ldots, x_n; y)$ such that $y = f(x_1, \ldots, x_n)$ after execution of P

Theorem

recursive functions are WHILE computable

Proof

- minimization $\quad f(y_1, \ldots, y_n) = (\mu\, x)\, (g(x, y_1, \ldots, y_n) = 0)$

Definition

function $f \colon \mathbb{N}^n \to \mathbb{N}$ is WHILE computable if $\exists$ WHILE program $P(x_1, \ldots, x_n; y)$ such that $y = f(x_1, \ldots, x_n)$ after execution of P

Theorem

recursive functions are WHILE computable

Proof

- minimization $\quad f(y_1, \ldots, y_n) = (\mu\, x)\,(g(x, y_1, \ldots, y_n) = 0)$

$$
\begin{array}{l}
x := 0\,;\ P_g(x, y_1, \ldots, y_n; z)\,; \\
\text{WHILE } z > 0 \text{ DO} \\
\quad x{+}{+}\,; \\
\quad P_g(x, y_1, \ldots, y_n; z) \\
\text{OD}
\end{array}
$$

Remark

not every WHILE computable function is recursive

Remark

not every WHILE computable function is recursive

Example

program $P(x;y)$:

$$
\begin{array}{l}
y := 0\,; \\
w := x\,; \\
\text{WHILE } w > 0 \text{ DO} \\
\quad y{+}{+}\,; \\
\quad P_{\times}(y,y;z)\,;\ P_{\dot-}(x,z;u)\,; \\
\quad P_{\dot-}(z,x;v)\,;\ P_{+}(u,v;w) \\
\text{OD}
\end{array}
$$

Remark

not every WHILE computable function is recursive

Example

program $P(x; y)$:

$y := 0$;

$w := x$;

WHILE $w > 0$ DO

$y{+}{+}$;

$P_\times(y, y; z)$; $P_{\dot{-}}(x, z; u)$; $\qquad u = x \dot{-} y^2$

$P_{\dot{-}}(z, x; v)$; $P_+(u, v; w)$

OD

Remark

not every WHILE computable function is recursive

Example

program $P(x; y)$:

$y := 0$;

$w := x$;

WHILE $w > 0$ DO

$y{+}{+}$;

$P_\times(y, y; z)$; $P_{\dot{-}}(x, z; u)$; $\qquad u = x \dot{-} y^2$

$P_{\dot{-}}(z, x; v)$; $P_+(u, v; w)$ $\qquad w = (x \dot{-} y^2) + (y^2 \dot{-} x)$

OD

Remark

not every WHILE computable function is recursive

Example

program $P(x;y)$:

$y := 0$;

$w := x$;

WHILE $w > 0$ DO

$y{+}{+}$;

$P_\times(y,y;z)$; $P_{\dot{-}}(x,z;u)$; $\qquad u = x \dot{-} y^2$

$P_{\dot{-}}(z,x;v)$; $P_+(u,v;w)$ $\qquad w = (x \dot{-} y^2) + (y^2 \dot{-} x) = |x - y^2|$

OD

Remark

not every WHILE computable function is recursive

Example

program $P(x; y)$:

$y := 0\,;$

$w := x\,;$

WHILE $w > 0$ DO

$y{+}{+}\,;$

$P_\times(y, y; z)\,;\; P_{\dot{-}}(x, z; u)\,;$ $\qquad u = x \dot{-} y^2$

$P_{\dot{-}}(z, x; v)\,;\; P_+(u, v; w)$ $\qquad w = (x \dot{-} y^2) + (y^2 \dot{-} x) = |x - y^2|$

OD

computes partial function $\sqrt{x} = (\mu\, y)\,(x = y^2)$

Outline

1. Summary of Previous Lecture
2. Recursive Functions
3. While Programs
4. **Partial Recursive Functions**
5. Normal Form Theorem
6. Summary

Definition

class PA of partial recursive functions is smallest class of partial functions that contains all initial functions and is closed under composition, primitive recursion, and unbounded minimization:

$$(\mu\, i)\,(f(i,\vec{y}) = 0) = \min\,\{\, i \mid f(i,\vec{y}) = 0 \text{ and } f(j,\vec{y}) > 0 \text{ for all } j < i \,\}$$

belongs to PA whenever $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ belongs to PA

Definition

class PA of partial recursive functions is smallest class of partial functions that contains all initial functions and is closed under composition, primitive recursion, and unbounded minimization:

$$(\mu\, i)\,(f(i,\vec{y}) = 0) = \min\{i \mid f(i,\vec{y}) = 0 \text{ and } f(j,\vec{y}) > 0 \text{ for all } j < i\}$$

belongs to PA whenever $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ belongs to PA

Definition (semantics)

partial recursive expressions are evaluated according to call–by–value semantics

Definition

class PA of partial recursive functions is smallest class of partial functions that contains all initial functions and is closed under composition, primitive recursion, and unbounded minimization:

$$(\mu\, i)\, (f(i,\vec{y}) = 0) = \min\{i \mid f(i,\vec{y}) = 0 \text{ and } f(j,\vec{y}) > 0 \text{ for all } j < i\}$$

belongs to PA whenever $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ belongs to PA

Definition (semantics)

partial recursive expressions are evaluated according to call–by–value semantics

Example

function $\varphi(x) = z((\mu\, i)\, (i + x = 0))$ is undefined for $x > 0$

Theorem

partial recursive functions are WHILE computable

Theorem

partial recursive functions are WHILE computable

Proof

- $\ldots$
- minimization $\quad f(y_1, \ldots, y_n) = (\mu\, x)\, (g(x, y_1, \ldots, y_n) = 0)$

Theorem

partial recursive functions are WHILE computable

Proof

- $\cdots$
- minimization $\quad f(y_1,\dots,y_n) = (\mu\, x)\,(g(x,y_1,\dots,y_n) = 0)$

$$x := 0\,;\; P_g(x,y_1,\dots,y_n;z)\,;$$
$$\text{WHILE } z > 0 \text{ DO}$$
$$\quad x{+}{+}\,;$$
$$\quad P_g(x,y_1,\dots,y_n;z)$$
$$\text{OD}$$

Theorem

WHILE computable functions are partial recursive

Theorem

WHILE computable functions are partial recursive

Corollary

function φ is partial recursive $\iff$ φ is WHILE computable

Theorem

WHILE computable functions are partial recursive

Corollary

function φ is partial recursive $\iff$ φ is WHILE computable

Notation

- $\varphi(x_1, \dots, x_n)\uparrow$ if $\varphi(x_1, \dots, x_n)$ is undefined

Theorem

WHILE computable functions are partial recursive

Corollary

function φ is partial recursive $\iff$ φ is WHILE computable

Notation

- $\varphi(x_1, \ldots, x_n)\uparrow$ if $\varphi(x_1, \ldots, x_n)$ is undefined
- $\varphi(x_1, \ldots, x_n)\downarrow$ if $\varphi(x_1, \ldots, x_n)$ is defined

Theorem

WHILE computable functions are partial recursive

Corollary

function φ is partial recursive $\iff$ φ is WHILE computable

Notation

- $\varphi(x_1,\ldots,x_n)\uparrow$ if $\varphi(x_1,\ldots,x_n)$ is undefined
- $\varphi(x_1,\ldots,x_n)\downarrow$ if $\varphi(x_1,\ldots,x_n)$ is defined
- $\varphi \simeq \psi$ if for all $x_1,\ldots,x_n \in \mathbb{N}$ either

  ① $\varphi(x_1,\ldots,x_n)\uparrow$ and $\psi(x_1,\ldots,x_n)\uparrow$

Theorem

WHILE computable functions are partial recursive

Corollary

function φ is partial recursive $\iff$ φ is WHILE computable

Notation

- $\varphi(x_1,\dots,x_n)\uparrow$ if $\varphi(x_1,\dots,x_n)$ is undefined
- $\varphi(x_1,\dots,x_n)\downarrow$ if $\varphi(x_1,\dots,x_n)$ is defined
- $\varphi \simeq \psi$ if for all $x_1,\dots,x_n \in \mathbb{N}$ either
  ① $\varphi(x_1,\dots,x_n)\uparrow$ and $\psi(x_1,\dots,x_n)\uparrow$ or
  ② $\varphi(x_1,\dots,x_n)\downarrow$ and $\psi(x_1,\dots,x_n)\downarrow$ and $\varphi(x_1,\dots,x_n) = \psi(x_1,\dots,x_n)$

Outline

1. Summary of Previous Lecture
2. Recursive Functions
3. While Programs
4. Partial Recursive Functions
5. **Normal Form Theorem**
6. Summary

Definition

index $\ulcorner f \urcorner \in \mathbb{N}$ of derivation of partial recursive function f is defined inductively:

- $\ulcorner \mathrm{z} \urcorner = \langle 0 \rangle$
- $\ulcorner \mathrm{s} \urcorner = \langle 1 \rangle$
- $\ulcorner \pi_i^n \urcorner = \langle 2, n, i \rangle$
- $\ulcorner f \urcorner = \langle 3, \ulcorner g \urcorner, \ulcorner h_1 \urcorner, \dots, \ulcorner h_m \urcorner \rangle$ if f is obtained by composing g and $h_1, \dots, h_m$
- $\ulcorner f \urcorner = \langle 4, \ulcorner g \urcorner, \ulcorner h \urcorner \rangle$ if f is obtained by primitive recursion from g and h

Definition

index $\ulcorner f \urcorner \in \mathbb{N}$ of derivation of partial recursive function f is defined inductively:

- $\ulcorner \mathsf{z} \urcorner = \langle 0 \rangle$
- $\ulcorner \mathsf{s} \urcorner = \langle 1 \rangle$
- $\ulcorner \pi^n_i \urcorner = \langle 2, n, i \rangle$
- $\ulcorner f \urcorner = \langle 3, \ulcorner g \urcorner, \ulcorner h_1 \urcorner, \dots, \ulcorner h_m \urcorner \rangle$ if f is obtained by composing g and $h_1, \dots, h_m$
- $\ulcorner f \urcorner = \langle 4, \ulcorner g \urcorner, \ulcorner h \urcorner \rangle$ if f is obtained by primitive recursion from g and h
- $\ulcorner f \urcorner = \langle 5, \ulcorner g \urcorner \rangle$ if f is obtained by minimizing g

Definition

index $\ulcorner f \urcorner \in \mathbb{N}$ of derivation of partial recursive function f is defined inductively:

- $\ulcorner \mathsf{z} \urcorner = \langle 0 \rangle$
- $\ulcorner \mathsf{s} \urcorner = \langle 1 \rangle$
- $\ulcorner \pi^n_i \urcorner = \langle 2, n, i \rangle$
- $\ulcorner f \urcorner = \langle 3, \ulcorner g \urcorner, \ulcorner h_1 \urcorner, \dots, \ulcorner h_m \urcorner \rangle$ if f is obtained by composing g and $h_1, \dots, h_m$
- $\ulcorner f \urcorner = \langle 4, \ulcorner g \urcorner, \ulcorner h \urcorner \rangle$ if f is obtained by primitive recursion from g and h
- $\ulcorner f \urcorner = \langle 5, \ulcorner g \urcorner \rangle$ if f is obtained by minimizing g

Remark (key insight)

from index we can reconstruct function

Kleene's Normal Form Theorem

$\exists$ primitive recursive function u

Kleene's Normal Form Theorem

$\exists$ primitive recursive function u $\forall\, n \geqslant 1$ $\exists$ primitive recursive predicate T_n

Kleene's Normal Form Theorem

$\exists$ primitive recursive function u $\;\forall\, n \geqslant 1\;$ $\exists$ primitive recursive predicate T_n

$\forall$ partial recursive function $\varphi \colon \mathbb{N}^n \to \mathbb{N}$

$$\varphi(x_1,\dots,x_n) \simeq \mathsf{u}((\mu\, y)\, \mathsf{T}_n(\ulcorner \varphi \urcorner, x_1,\dots,x_n,y))$$

Kleene's Normal Form Theorem

$\exists$ primitive recursive function u $\forall n \geqslant 1$ $\exists$ primitive recursive predicate T_n
$\forall$ partial recursive function $\varphi\colon \mathbb{N}^n \to \mathbb{N}$

$$\varphi(x_1,\dots,x_n) \simeq \mathrm{u}((\mu\, y)\, \mathrm{T}_n(\ulcorner\varphi\urcorner, x_1,\dots,x_n,y))$$

Corollary

1. every partial recursive function can be defined using one application of minimization

Kleene's Normal Form Theorem

$\exists$ primitive recursive function u $\forall\, n \geqslant 1$ $\exists$ primitive recursive predicate T_n

$\forall$ partial recursive function $\varphi\colon \mathbb{N}^n \to \mathbb{N}$

$$\varphi(x_1,\dots,x_n) \simeq \mathrm{u}((\mu\, y)\, \mathrm{T}_n(\ulcorner\varphi\urcorner, x_1,\dots,x_n,y))$$

Corollary

1. every partial recursive function can be defined using one application of minimization
2. partial recursiveness and recursiveness coincide for total functions

Outline

1. **Summary of Previous Lecture**
2. **Recursive Functions**
3. **While Programs**
4. **Partial Recursive Functions**
5. **Normal Form Theorem**
6. **Summary**

Important Concepts

- Cantor pairing function
- Gödel's β function
- index
- Kleene's normal form theorem
- PA
- partial recursive function
- WHILE computable
- WHILE program

Important Concepts

- Cantor pairing function
- Gödel's β function
- index
- Kleene's normal form theorem
- PA
- partial recursive function
- WHILE computable
- WHILE program

homework for October 30