universität
innsbruck

WS 2023 lecture 2

Computability Theory

Aart Middeldorp

Outline

Definition

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

- zero $\mathsf{z}(x) = 0$
- successor $\mathsf{s}(x) = x + 1$
- projection $\pi_i^n(x_1,\dots,x_n) = x_i$ for all $n \geqslant 1$ and $1 \leqslant i \leqslant n$

and is closed under composition

- $f(\vec{x}) = g(h_1(\vec{x}),\dots,h_m(\vec{x})) \in \mathrm{PR}$ for all $g\colon \mathbb{N}^m \to \mathbb{N} \in \mathrm{PR}$ and $h_1,\dots,h_m\colon \mathbb{N}^n \to \mathbb{N} \in \mathrm{PR}$

and primitive recursion

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

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

belongs to PR for all $g\colon \mathbb{N}^n \to \mathbb{N} \in \mathrm{PR}$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N} \in \mathrm{PR}$

Definition

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

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

is primitive recursive

Lemma

PR is closed under iteration, case analysis and bounded minimization

Definitions

- pairing function $\pi(x,y) = 2^x(2y+1) \mathbin{\dot{-}} 1$
- extraction functions

$$\pi_1(z) = (\mu\, x \leqslant z)\,(\exists\, y \leqslant z)\,\big[\, z = \pi(x,y) \,\big] \qquad \pi_2(z) = (\mu\, y \leqslant z)\,(\exists\, x \leqslant z)\,\big[\, z = \pi(x,y) \,\big]$$

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. Gödel Numbering
3. Course–of–Values Recursion
4. Ackermann Function
5. Diagonalization
6. Total Recursive Functions
7. Summary

Definitions

- Gödel number of sequence $x_1, \ldots, x_n$: $\langle x_1, \ldots, x_n \rangle = \mathsf{p}_0^n \times \mathsf{p}_1^{x_1} \times \cdots \times \mathsf{p}_n^{x_n}$
- $(x)_i = (\mu j < x)\, \neg(\mathsf{p}_i^{j+1} \mid x)$
- $\mathsf{len}(x) = (x)_0$
- $\mathsf{seq}(x) \iff x > 0 \wedge (\forall i \leqslant x) \left[(x)_i \neq 0 \implies i \leqslant \mathsf{len}(x) \right]$
- $x \,;\, y = \mathsf{p}_0^{\mathsf{len}(x)+\mathsf{len}(y)} \times \prod_{i<\mathsf{len}(x)} \mathsf{p}_{i+1}^{(x)_{i+1}} \times \prod_{i<\mathsf{len}(y)} \mathsf{p}_{\mathsf{len}(x)+i+1}^{(y)_{i+1}}$

Lemma

- $x = \langle x_1, \ldots, x_n \rangle \implies \mathsf{len}(x) = n \wedge (x)_i = x_i$ for all $1 \leqslant i \leqslant n$
- $\mathsf{seq}(x) \wedge \mathsf{len}(x) = n \implies x = \langle (x)_1, \ldots, (x)_n \rangle$
- $x = \langle x_1, \ldots, x_n \rangle \wedge y = \langle y_1, \ldots, y_m \rangle \implies x \,;\, y = \langle x_1, \ldots, x_n, y_1, \ldots, y_m \rangle$

Outline

1. Summary of Previous Lecture
2. Gödel Numbering
3. Course–of–Values Recursion
4. Ackermann Function
5. Diagonalization
6. Total Recursive Functions
7. Summary

Definition

if $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ then $\tilde{f}(x,\vec{y}) = \langle f(0,\vec{y}),\ldots,f(x,\vec{y})\rangle$

Lemma

if $g\colon \mathbb{N}^{n} \to \mathbb{N}$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$ are primitive recursive then so is $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined as

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

Proof

$f(x,\vec{y}) = (\tilde{f}(x,\vec{y}))_{x+1}$ and $\tilde{f}$ is primitive recursive:

$$\begin{aligned}\tilde{f}(0,\vec{y}) &= \langle f(0,\vec{y})\rangle = \langle g(\vec{y})\rangle\\ \tilde{f}(x+1,\vec{y}) &= \tilde{f}(x,\vec{y}) \mathbin{;} \langle f(x+1,\vec{y})\rangle = \tilde{f}(x,\vec{y}) \mathbin{;} \langle h(\tilde{f}(x,\vec{y}),x,\vec{y})\rangle = h'(\tilde{f}(x,\vec{y}),x,\vec{y})\end{aligned}$$

with $h'(x,y,\vec{z}) = x \mathbin{;} \langle h(x,y,\vec{z})\rangle$

Outline

1. Summary of Previous Lecture
2. Gödel Numbering
3. Course–of–Values Recursion
4. **Ackermann Function**
5. Diagonalization
6. Total Recursive Functions
7. Summary

Definition

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

Example

- $\mathrm{ack}(1,3) = 5$
- $\mathrm{ack}(2,2) = 7$
- $\mathrm{ack}(3,4) = 125$
- $\mathrm{ack}(4,1) = 65533$

Lemma

Ackermann function is well-defined

Example (cont'd)

- $\mathrm{ack}(4,3) = 2^{2^{65533}} - 3$

Remarks

- Ackermann function grows very fast
- Ackermann function is used as compiler benchmark
- inverse of Ackermann function appears in complexity analysis of certain algorithms

Lemma

① $\mathrm{ack}(x,y) > y$

② $\mathrm{ack}(x,y+1) > \mathrm{ack}(x,y)$

③ $\mathrm{ack}(x+1,y) \geqslant \mathrm{ack}(x,y+1)$

④ $\mathrm{ack}(x+2,y) > \mathrm{ack}(x,2y)$

skip proofs

Lemma ①

- $\mathsf{ack}(x,y) > y$

Proof

induction on x

- $\mathsf{ack}(0,y) = y+1 > y$
- induction on y

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

Lemma ②

- $\mathsf{ack}(x,y+1) > \mathsf{ack}(x,y)$

Proof

induction on x

$$\begin{aligned} \mathsf{ack}(0,y+1) &= y+2 > y+1 = \mathsf{ack}(0,y) \\ \mathsf{ack}(x+1,y+1) &= \mathsf{ack}(x,\mathsf{ack}(x+1,y)) > \mathsf{ack}(x+1,y) \end{aligned}$$

Lemma ③

- $\mathsf{ack}(x+1,y) \geqslant \mathsf{ack}(x,y+1) > \mathsf{ack}(x,y)$

Proof

induction on x

- $\mathsf{ack}(1,y) \geqslant y+2 = \mathsf{ack}(0,y+1)$ by induction on y

$$\begin{aligned} \mathsf{ack}(1,0) &= \mathsf{ack}(0,1) = 2 \\ \mathsf{ack}(1,y+1) &= \mathsf{ack}(0,\mathsf{ack}(1,y)) \geqslant \mathsf{ack}(0,y+2) \end{aligned}$$

- induction on y

$$\begin{aligned} \mathsf{ack}(x+2,0) &= \mathsf{ack}(x+1,1) \\ \mathsf{ack}(x+2,y+1) &= \mathsf{ack}(x+1,\mathsf{ack}(x+2,y)) \geqslant \mathsf{ack}(x+1,\mathsf{ack}(x+1,y+1)) \\ &\geqslant \mathsf{ack}(x+1,\mathsf{ack}(x,y+2)) > \mathsf{ack}(x+1,y+2) \end{aligned}$$

Lemma ④

- $\mathsf{ack}(x+2,y) > \mathsf{ack}(x,2y)$

Proof

induction on x

- $\mathsf{ack}(2,y) > \mathsf{ack}(0,2y)$ by induction on y

$$\begin{aligned} \mathsf{ack}(2,0) &\geqslant \mathsf{ack}(1,1) \geqslant \mathsf{ack}(0,2) > \mathsf{ack}(0,1) > \mathsf{ack}(0,0) \\ \mathsf{ack}(2,y+1) &= \mathsf{ack}(1,\mathsf{ack}(2,y)) > \mathsf{ack}(1,2y+1) \geqslant \mathsf{ack}(0,2y+2) \end{aligned}$$

- induction on y

$$\begin{aligned} \mathsf{ack}(x+3,0) &\geqslant \mathsf{ack}(x+2,1) \geqslant \mathsf{ack}(x+1,2) > \mathsf{ack}(x+1,0) \\ \mathsf{ack}(x+3,y+1) &= \mathsf{ack}(x+2,\mathsf{ack}(x+3,y)) > \mathsf{ack}(x+2,\mathsf{ack}(x+1,2y)) \\ &\geqslant \mathsf{ack}(x+2,\mathsf{ack}(x,2y+1)) > \mathsf{ack}(x+2,2y+1) \\ &\geqslant \mathsf{ack}(x+1,2y+2) \end{aligned}$$

Lemma

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

$$f(x_1,\ldots,x_n) < \text{ack}(c, \max\{x_1,\ldots,x_n\})$$

Proof

induction on definition of primitive recursive functions

- initial functions

  $z(x) = 0 < x+1 = \text{ack}(0,x)$

  $s(x) = x+1 = \text{ack}(0,x) < \text{ack}(1,x)$

  $\pi_i^n(x_1,\ldots,x_n) = x_i \leqslant \max\{x_1,\ldots,x_n\} < \text{ack}(0,\max\{x_1,\ldots,x_n\})$

Lemma

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

$$f(x_1,\ldots,x_n) < \text{ack}(c, \max\{x_1,\ldots,x_n\})$$

Proof (cont'd)

induction on definition of primitive recursive functions

- composition $f(\vec{x}) = g(h_1(\vec{x}),\ldots,h_m(\vec{x}))$

  $g(\vec{y}) < \text{ack}(c_0, \max\{\vec{y}\})$ and $h_i(\vec{x}) < \text{ack}(c_i, \max\{\vec{x}\})$ for all $1 \leqslant i \leqslant m$

  $c' = \max\{c_0+1, c_1,\ldots,c_m\}$

$$\begin{aligned} f(\vec{x}) &< \text{ack}(c_0, \max\{h_1(\vec{x}),\ldots,h_m(\vec{x})\}) \\ &< \text{ack}(c_0, \max\{\text{ack}(c_1,\max\{\vec{x}\}),\ldots,\text{ack}(c_m,\max\{\vec{x}\})\}) \\ &= \text{ack}(c_0, \text{ack}(\max\{c_1,\ldots,c_m\}, \max\{\vec{x}\})) \\ &\leqslant \text{ack}(c'-1, \text{ack}(c', \max\{\vec{x}\})) = \text{ack}(c', \max\{\vec{x}\}+1) \leqslant \text{ack}(c'+1, \max\{\vec{x}\}) \end{aligned}$$

Lemma

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

$$f(x_1,\ldots,x_n) < \text{ack}(c, \max\{x_1,\ldots,x_n\})$$

Proof (cont'd)

induction on definition of primitive recursive functions

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

  $g(\vec{y}) < \text{ack}(c_1, \max\{\vec{y}\})$ and $h(a,b,\vec{y}) < \text{ack}(c_2, \max\{a,b,\vec{y}\})$

  $c' = \max\{c_1, c_2+1\}$

  $f(x,\vec{y}) < \text{ack}(c', x+\max\{\vec{y}\})$ by induction on x

$$f(x,\vec{y}) < \text{ack}(c', 2\cdot\max\{x,\vec{y}\}) < \text{ack}(c'+2, \max\{x,\vec{y}\})$$

Theorem

Ackermann function is not primitive recursive

Proof

if Ackermann function is primitive recursive then

$$\text{ack}(x,y) < \text{ack}(c, \max\{x,y\})$$

for some constant c and thus

$$\text{ack}(c,c) < \text{ack}(c, \max\{c,c\}) = \text{ack}(c,c)$$ ↯

Outline

1. **Summary of Previous Lecture**
2. **Gödel Numbering**
3. **Course–of–Values Recursion**
4. **Ackermann Function**
5. **Diagonalization**
6. **Total Recursive Functions**
7. **Summary**

Definition

index $\ulcorner f \urcorner \in \mathbb{N}$ of derivation of primitive 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

Examples

- $\ulcorner + \urcorner = \langle 4, \langle 2, 1, 1 \rangle, \langle 3, \langle 1 \rangle, \langle 2, 3, 1 \rangle \rangle \rangle$
- $397065375000 = 2^3 \times 3^3 \times 5^6 \times 7^6 = \langle 3, \langle 1 \rangle, \langle 1 \rangle \rangle = \ulcorner \mathsf{s} \circ \mathsf{s} \urcorner$

Remark (key insight)

from index we can reconstruct function (details in later lecture)

Corollary

$\exists$ computable list $f_0, f_1, f_2, f_3, \dots$ of unary primitive recursive functions

Diagonalization

$$\begin{array}{ccccc} f_0(0) & f_0(1) & f_0(2) & f_0(3) & \cdots \\ f_1(0) & f_1(1) & f_1(2) & f_1(3) & \cdots \\ f_2(0) & f_2(1) & f_2(2) & f_2(3) & \cdots \\ f_3(0) & f_3(1) & f_3(2) & f_3(3) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

$$g(x) = f_x(x) + 1$$

Corollary

function g is computable but not primitive recursive

Outline

1. **Summary of Previous Lecture**
2. **Gödel Numbering**
3. **Course–of–Values Recursion**
4. **Ackermann Function**
5. **Diagonalization**
6. **Total Recursive Functions**
7. **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 (unbounded) minimization:

$$(\mu\, i)\; P(i,\vec{y}) = \min\{i \mid P(i,\vec{y})\} \in \mathrm{R}$$

for all $P\colon \mathbb{N}^{n+1} \to \mathbb{B}$ such that $\chi_P \in \mathrm{R}$ and $\forall\, \vec{y}\; \exists\, x\; P(x,\vec{y})$

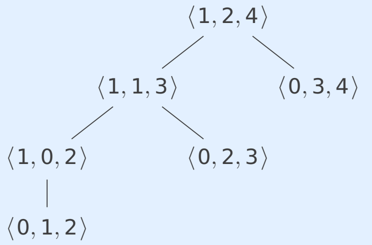
Theorem

Ackermann function is recursive

Representing Computations

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

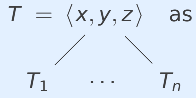
$\mathrm{ack}(1,2) = 4$:



$\langle x,y,z\rangle$ denotes $\mathrm{ack}(x,y) = z$

Definition

encode computation tree



as number $\ulcorner T \urcorner = \langle \langle x,y,z\rangle, \ulcorner T_1 \urcorner, \dots, \ulcorner T_n \urcorner \rangle$

Lemma

$\exists$ primitive recursive predicate $P\colon \mathbb{N} \to \mathbb{B}$ such that

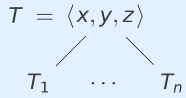
$$P(x) \iff x \text{ encodes correct computation tree}$$

Corollary

- $\mathrm{ack}(x,y) = z \iff \exists\, t$ such that $P(t)$ and $(t)_1 = \langle x,y,z\rangle$
- $\mathrm{ack}(x,y) = ((\mu\, t)\; (P(t) \wedge (t)_{1,1} = x \wedge (t)_{1,2} = y))_{1,3}$

Proof

- $P(x) \iff \mathrm{seq}(x) \wedge \mathrm{seq}((x)_1) \wedge \mathrm{len}((x)_1) = 3 \wedge (A(x) \vee B(x) \vee C(x))$
- $A(x) \iff \mathrm{len}(x) = 1 \wedge (x)_{1,1} = 0 \wedge (x)_{1,3} = \mathrm{s}((x)_{1,2})$
- $B(x) \iff \mathrm{len}(x) = 2 \wedge (x)_{1,1} > 0 \wedge (x)_{1,2} = 0 \wedge (x)_{2,1,1} = \mathrm{p}((x)_{1,1}) \wedge (x)_{2,1,2} = 1$ $\wedge\; (x)_{2,1,3} = (x)_{1,3} \wedge P((x)_2)$
- $C(x) \iff \mathrm{len}(x) = 3 \wedge (x)_{1,1} > 0 \wedge (x)_{1,2} > 0 \wedge (x)_{2,1,1} = (x)_{1,1} \wedge (x)_{2,1,2} = \mathrm{p}((x)_{1,2})$ $\wedge\; (x)_{3,1,1} = \mathrm{p}((x)_{1,1}) \wedge (x)_{3,1,2} = (x)_{2,1,3} \wedge (x)_{3,1,3} = (x)_{1,3} \wedge P((x)_2) \wedge P((x)_3)$



$\ulcorner T \urcorner = \langle \langle x,y,z\rangle, \ulcorner T_1 \urcorner, \dots, \ulcorner T_n \urcorner \rangle$

Outline

1. **Summary of Previous Lecture**
2. **Gödel Numbering**
3. **Course–of–Values Recursion**
4. **Ackermann Function**
5. **Diagonalization**
6. **Total Recursive Functions**
7. **Summary**

Important Concepts

- Ackermann function
- course–of–values recursion
- diagonalization
- Gödel numbering
- index
- R
- recursive function
- (unbounded) minimization

homework for October 16