Complexity Theory

Lection 6:

- $\mathcal{D}$ - subject area (objects, properly built functions, ...)
- $\nu : (D) \to \mathcal{N}$ - coding into unique natural number Actually anything can be coded using this numeration.
- there are lots of Godel numerations ($\nu(x_1, ..., x_n) = 2^{\nu(x_1)} ... p_n^{\nu(x_n)}$)
- arithmetization of arbitrary theory (transposition from theory objects to natural numbers)
- partial function $f : \mathbb{N}^n \not\to \mathbb{N}, n \in \mathbb{N}$, is (algorithmically) computable $\iff$ there is such Turing machine $M : M(x_1, ..., x_n) \simeq f(x_1, ..., x_n), \forall x_1, ..., x_n \in \mathbb{N}$ (Turing thesis)
- set (algorithmically) computable functions coincides to set of partially-recursive functions (Church thesis)
- Turing Machines numeration $M_0, M_1, ...$
- numeration of every (algorithmic) computable functions $\varphi_0, \varphi_1, ...$

Example of computable function:

$$f(n) = 1, \text{ if there would be colony on Moon}$$

$$f(n) = 0, \text{ if there would be no colony on Moon}$$

subjunctive algorithm that proves computability of this function is to wait the predefined set time.

Theorem about uncomputable function

Uncomputable function defined everywhere - exists.

Proof

- $\varphi_0^1, \varphi_1^1, ...$ - all computable functions (1)
- $$f(n) = \begin{cases} \varphi_n^1(n) + 1, & \text{if } \varphi_n^1(n) \neq \bot \\ 0, & \text{if } \varphi_n^1(n) = \bot \ \ (\nexists \varphi_n^1) \end{cases} \quad , \quad n \in \mathbb{N}$$
- $\forall n \in \mathbb{N} f \not\simeq \varphi_n^1 : f(n) \not\simeq \varphi_n^1(n)$
- diagonalization method (Kantor)

Theorem about parametrization

For arbitrary countable function $f(x, y)$ exists everywhere defined countable function $k(x)$ that $f(x, y) = \varphi_{k(x)}(y)$ for arbitrary Godel Numeration $\varphi_0, \varphi_1, ...$ unary countable functions.

Proof

- $f(x, y)$ is countable $\Rightarrow \exists$ TM $M : M(x, y) \simeq f(x, y)$
- $\forall a \in \mathbb{N} \exists$ TM $M_a : M_a(y) \simeq f(a, y)$
- composition of Turing Machines: add argument a at input (additional) tape and start TM M
- number of TM M_a is $k(a)$ value

Consequence Number of function $k(x)$ depends only on parameter x.

---

s_n^m **Kleene theorem (s-m-n Theorem)**

Theorem For arbitrary countable funcions' Godel Numeration exists such a primitiv recursive function $s : \mathbb{N}^2 \to \mathbb{N}$ (2), that for arbitrary Godel number $p \in \mathbb{N}$ of some partial function of two variables next Kleene equality is true: $\varphi_{s(p,x)}(y) \simeq \varphi_p(x, y)$ for every natural number $x, y \in \mathbb{N}$.

s_n^m **Kleene theorem (s-m-n theorem, parametrization theorem)** For arbitrary natural numbers $m, n > 0$ and arbitrary godel numeration of countable functions exists such a primitiv recursive function $s_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ that for arbitrary Godel number $p \in \mathbb{N}$ of some partial function of $m + n$ arguments Kleene equality is true: [_{s_n^m(p, x_1, ..., x_m)}(y_1, ..., y_n) _p(x_1, ..., x_m, y_1, ..., y_n)] for every natural number $x_1, ..., x_m, y_1, ..., y_n \in \mathbb{N}$.

Universal function

For arbitrary set of partial functions $\mathcal{H} \subseteq \mathcal{F}_n$ of n variables, $n \in \mathbb{N}_0$, function $f \in \mathcal{F}_{n+1}$ of variables $n + 1$ is called universal function of set of functions $\mathcal{H}$, if the following two conditions are true: - for arbitrary number $c \in \mathbb{N}_0$ function $f(c, \cdot)$ of variables n is in set of functions $\mathcal{H}$ - for arbitrary function h of set of functions $\mathcal{H}$ exists such a number $c \in \mathbb{N}_0$, that $h(x_1, ..., x_n) \simeq f(c, x_1, ..., x_n)$ for arbitrary values $x_1, ..., x_n \in \mathbb{N}_0$.

Algorithm to compute universal function for a set is called universal.

Theorem (about numeration) For arbitrary number $n \in \mathbb{N}_0$ exists such universal function of set of all partial computable functions $\mathcal{H}_n \subseteq \mathcal{F}_n$ of n variables.

Proof

- let $f(y, x_1, ..., x_n) \simeq \varphi_y(x_1, ..., x_n)$ for arbitrary numbers $y, x_1, ..., x_n \in \mathbb{N}_0$
- by value $y \in \mathbb{N}_0$ find algorithm of computing function φ_y and compute value $\varphi_y(x_1, ..., x_n)$ using this algorithm
- $\Rightarrow$ function f is computable

Theorem For arbitrary number $n \in \mathbb{N}_0$ there is no such universal funtion of set of defined everywhere computable functions $\mathcal{H}_n^{tot} \subseteq \mathcal{F}_n^{tot} \subset \mathcal{F}_n$ of n variables.

Proof

- let universal function f of set of functions $\mathcal{H}_n^{tot}$: $f(y, x_1, ..., x_n) = \varphi_y(x_1, ..., x_n)$ for arbitrary numbers $y, x_1, ..., x_n \in \mathbb{N}_0$
- let $h(x_1, ..., x_n) = f(x_1, x_1, ..., x_n) + 1$ for arbitrary numbers $x_1, ..., x_n \in \mathbb{N}_0$
- $\Rightarrow h \in \mathcal{H}_n^{tot} \Rightarrow \exists c \in \mathbb{N}_0 f(c, x_1, ..., x_n) = h(x_1, ..., x_n)$ for arbitrary numbers $x_1, ..., x_n \in \mathbb{N}_0$
- on one side $h(c, ..., c) = f(c, ..., c)$ but $h(c, ..., c) = f(c, ..., c) + 1$ by definition of $h \Rightarrow$ *contradiction.*

- every universal function of unary computable functions set defines numeration $f(x, y) = \varphi_x(y)$
- binary function U is called **main universal function (main numeration)** if for any binary computable function h exists such a defined everywhere computable unary funtion g that $h(x, y) = U(g(x), y)$ for any numbers $x, y \in \mathbb{N}_0$
- $\Rightarrow$ exists main universal function of set of all unary computable functions
- $\Rightarrow U_1(x, y) = U_2(c_1(x), y)$ and $U_2(x, y) = U_1(c_2(x), y)$ (**theorem about main numerations' ismorphism**)
- operations on computable functions $\Leftrightarrow$ operations on their indexes

---

Theorem about motionless point

For arbitrary Godel numeration $\varphi_0 . \varphi_1, ...$ of unary computable functions and arbitrary unary computable defined everywhere funtion f exists such natural number $n \in \mathbb{N}_0$ that $\varphi_n \simeq \varphi_{f(n)}$.

Proof

- consider such function $\varphi_{f(\varphi_x(x))}(y)$, $\varphi_{f(\varphi_x(x))}(y) \simeq \psi(f(\varphi_x(x)), y) \simeq g(x, y), \forall x, y \in \mathbb{N}_0$
- from s_n^m Kleene theorem follows that exists such unary defined everywhere function h that $\varphi_{f(\varphi_x(x))}(y) \simeq \varphi_{h(x)}(y), \forall x, y \in \mathbb{N}_0$

- let $h \simeq \varphi_m \Rightarrow \varphi_{f(\varphi_x(x))}(y) \simeq \varphi_{\varphi_m}(y), \forall x, y \in \mathbb{N}_0$
- let $\varphi_m(m) = n$ (defined everywhere) $\Rightarrow \varphi_{f(n)}(y) \simeq \varphi_n(y), \forall y \in \mathbb{N}_0$

Second theorem about recursion (Kleene, 1938) For arbitrary Godel numeration $\varphi_0 . \varphi_1, ...$ unary omputable functions and arbitrary binary partial computable function f exists such natural number $n \in \mathbb{N}_0$ that $\varphi_n(y) \simeq f(n, y)$ for all numbers $y \in \mathbb{n}_0$.

Consequence Let function h - $f(x, y) \simeq \varphi_{h(x)}(y)$ (s_n^m theorem). Let number m be motionless point of function h. From theorem about Rodger's motionless point follows second theorem about recursion ($n = m$, $\varphi_m(y) \simeq \varphi_{h(m)}(y) \simeq f(m, y)$ for all numbers $y \in \mathbb{N}_0$)

Consequence Let function f - for arbitrary algorithm $\mathcal{A}_x$ algorithm $\mathcal{A}_{f(x)}$ "prints description" of algorithm $\mathcal{A}_x$. Function f is computable $\Rightarrow$ by theorem about motionless point exists algorithm $\mathcal{A}$, that "prints own description".

Computable functions

Can UTM compute arbitrary function $\{0,1\}^* \to \{0,1\}^*$?

Theorem Exists uncountable function UC: $\{0,1\}^* \to \{0,1\}$

Proof

- TM numeration using set $\{0,1\}^*$, for arbitrary word $x \in \{0,1\}^*$ appropriate Turing Machine is marked M_x or $M_{\lceil x \rceil}$
- define

$$UC(x) = \begin{cases} 0, \text{ if } M_x(x) = 1 \\ 1, otherwise \end{cases} \quad \forall x \in \{0,1\}^*$$

- let $\exists$ TM $\widetilde{M} : \forall x \in \{0,1\}^* \widetilde{M}(x) = UC(x)$ $\widetilde{M}(\lfloor \widetilde{M} \rfloor) = ?$
- if $\widetilde{M}(\lfloor \widetilde{M} \rfloor) = 1$, then $UC(\lfloor \widetilde{M} \rfloor) = 0$ and vice versa

Modification of TM for recognition tasks

Recognition task $\Leftrightarrow$ defined everywhere function $\{0,1\}^* \to \{0,1\}$

Definition (multitape Turing Machine)

- $k \in \mathbb{N}^+$ number of tapes
- Γ Turing Machine alphabet
- $\# \in \Gamma$
- $\{0,1\}^*$ input alphabet
- Q nonempty finite set of internal states
- $q_0 \in Q$ initial state

- $q_{acc} \in Q$ final state, that accepts input word
- $q_{rej} \in Q, q_{acc} \neq q_{rej}$ final state that rejects input word
- $\delta : (Q \backslash \{q_{acc}, q_{rej}\}) \times \Gamma^k \nrightarrow Q \times \Gamma^{k-1} \times \{L, S, R\}^k$ partial function of transitions

Notion $q_{acc}, q_{rej} \in Q, q_{acc} \neq q_{rej}$, other ending configurations does not exist $(\Sigma = \{0,1\}, q_{acc} \equiv q_{accept} \equiv q_y \equiv q_{yes}, q_{rej} \equiv q_{reject} \equiv q_n \equiv q_{no})$

Definition Final configuration of TM is called positive (negative) if it's state is final state that accepts (rejects) input word.

Definition TM M input word x - accepts if $M(x) = 1$ (q_{acc}, positive configuration) - rejects if $M(x) = 0$ - not accepts if $M(x) = 0$ or $M(x) = \perp$ - not rejects if $M(x) = 1$ or $M(x) = \perp$

Language recognition

Definition TM M resolves (decides) language $L \subseteq \{0,1\}^*$ - if $x \in L$ then $M(x) = 1$ - if $x \notin L$ then $M(x) = 0$

Definition TM M recognizes language $L \subseteq \{0,1\}^*$ - if $x \in L$ then $M(x) = 1$ - if $x \notin L$ then $M(x) = 0$ or $M(x) = \perp$

Definition Languages - decidable (recursive) or semidecidable (recursively countable)

language $L(M)$ (L_M) of TM M - all word it accepts.

Definition Turing machines M_1 and M_2 are: - same if there exists such permutation of inner states and/or change of directions 'left' and 'right', otherwise - in principle different - equivalent if $M_1 = M_2$, $M_1 \simeq M_2$ - with one language if $L(M_1) = L(M_2)$

HALT problem

Define by binary representation of TM M and input word $x \in \{0,1\}^*$, will TM M stop on input word x. (decide language L_{HALT})

Theorem **HALT** task is unsolvable.

Proof

- let existance of M_{HALT}
- $M_{diag}(x) = M_{HALT}(x, x)$

- $$M^{co}(x) = \begin{cases} \text{cycle}, M_{diag}(x) = 1 \\ \text{stop}, M_{diag}(x) = 0 \end{cases}$$
- $M^{co}(\lfloor M^{co} \rfloor)$?

$HALT_\varepsilon$ problem

Define by binary representation of Turing Machine whether TM M will stop on empty input word (decide language L_{HALT_ε}).

Theorem Problem $HALT_\varepsilon$ is unsolvable.

Proof

- for arbitrary pair of TM $\widetilde{M}$ and input word x there exists TM $\widetilde{M}_x$
- if such TM exists, that solves $HALT_\varepsilon$ problem, then it solves $HALT$ problem
- **contradiction**

Rice's theorem

Numeric set $S \subseteq \mathbb{N}$ is called **invariant**, if representation of any two equivalent TM simultaneously is in or not in set S.

Examples

- all TM, that accepts input word 11
- all TM, that accepts at least one input word
- all TM, that never get hung up
- all TM, that stop after 15 tacts with input word 1
