Vyčíslitelnost a složitost

(bc. předmět KMI/VYSLO)

Petr Jančar

21. října 2025

Webová stránka předmětu

`http://phoenix.inf.upol.cz/~jancarp/VAS/vas.htm`

(na níž je i odkaz na stránku cvičení, které vede Mgr. Jiří Balun, Ph.D.).

Poznámky k textu. Tento text vzniká v průběhu kurzu v zimním semestru 2025/26 jako podpůrný materiál k přednáškám a cvičením. (Vzniká úpravou dřívější verze. Byť by se měl text dotknout všeho podstatného z přednášek a cvičení, jistě je nemůže nahradit.)

Text bude postupně přibývat, aktuálnost verze bude zřejmá z data uvedeného výše a také z počtu zahrnutých týdnů; text bude zhruba členěn po jednotlivých týdnech v semestru. Pro přehlednost zápis k jednotlivému týdnu začíná vždy na nové straně.

Jako základní učebnici ke kurzu lze uvést např. knihu

Sipser M. Introduction to the Theory of Computation. PWS Publishing Company, Boston, MA, 1997. ISBN 0-534-94728-X,

která již vyšla ve více vydáních. Pro teorii vyčíslitelnosti lze doporučit také např. knihu

Kozen D. Automata and Computability (Springer 1997),

která by z domény "upol" měla být přístupná v elektronické formě (pro osobní použití při studiu). V příslušné oblasti existuje ovšem široké spektrum učebnic a učebních materiálů, mnohé jsou i volně přístupné na webu. Některé materiály lze např. najít na stránkách starších běhů podobného kurzu doc. Jana Konečného.

V našem textu se speciálně budeme odkazovat i na pasáže pracovního materiálu autora textu, který v minulosti používal na VŠB-TU (a který je také přímo přístupný na webové stránce předmětu):

[JAN-TI] Jančar P.: Teoretická informatika (VŠB-TU Ostrava, 2010).

Týden 1

Začali jsme pár slovy o obsahu teorie vyčíslitelnosti a složitosti a o plánovaném běhu našeho kurzu (včetně návaznosti na teorii jazyků a automatů).

Upozornil jsem na první zápočtovou písemku v 6. výukovém týdnu semestru (který je 7. týdnem semestru kvůli svátku 28.10.) a druhou zápočtovou písemku v 12. týdnu (s možností oprav v posledním, 13. týdnu).

Problémy, algoritmy, Turingovy stroje.

Obsah přednášky a cvičení je pokryt částmi 6.1. a 6.2. textu [JAN-TI].

(Připomeňme, že odkazem [JAN-TI] míníme text *Jančar: Teoretická informatika*, který je uveden na první straně a který je přímo přístupný z webové stránky předmětu.)

Ujasnili jsme si technický význam pojmu *problém* (či *výpočetní problém*) v našem kontextu, speciálně pak pojmu *rozhodovací problém* (neboli *ANO/NE problém*), *algoritmicky (ne)řešitelný problém*, *(ne)rozhodnutelný problém*.

Problém P můžeme chápat jako trojici (IN, OUT, p), kde IN je množina vstupů, neboli instancí, OUT je množina výstupů a p je zobrazení typu $p : IN \to OUT$. V případě rozhodovacího problému máme $OUT = \{0, 1\}$, kde 0 je chápána jako FALSE a 1 jako TRUE.

Řekneme, že problém $P = (IN, OUT, p)$ je algoritmicky řešitelný, jestliže existuje algoritmus, který ho řeší, tj. algoritmus, který ke každému vstupu $w \in IN$ provede konečný výpočet a vydá problémem předepsaný výstup $p(w)$. V případě ANO/NE problému říkáme také "rozhodnutelný" místo "algoritmicky řešitelný".

Zmínili jsme např. problém třídění (či seřazení), který je algoritmicky řešitelný. Také jsme zmínili rozhodovací problémy SAT (splnitelnost booleovských formulí) a Eq-FA (ekvivalence konečných automatů), které jsou rozhodnutelné.

O problému Eq-CFG (ekvivalence bezkontextových gramatik) jsme si jen řekli, že je nerozhodnutelný. Později to dokážeme, teď nám ale nebylo jasné, jak dokázat neexistenci příslušného algoritmu. V prvé řadě si musíme pojem "algoritmus" více ujasnit.

Nejdříve jsme si připomněli, že pro řešení problémů algoritmy musíme jejich vstupy a výstupy nějak přirozeně kódovat, např. slovy (neboli řetězci) v nějaké konečné abecedě. Problém lze tak přirozeně chápat jako funkci $f : \Sigma^* \to \Sigma^*$ pro nějakou konečnou abecedu Σ (např. stačí $\Sigma = \{0, 1\}$, protože prvky jakékoli konečné abecedy lze jednoduše zakódovat dostatečně dlouhými řetězci bitů).

Funkce $f : \Sigma^* \to \Sigma^*$ je *algoritmicky vyčíslitelná* (někdy se též říká jen *vyčíslitelná*), jestliže existuje algoritmus, který ke každému $w \in \Sigma^*$ po konečném výpočtu (neboli běhu algoritmu) vydá $f(w)$.

Všimli jsme si souvislosti rozhodovacích problémů s jazyky: každý takový (ANO/NE) problém P můžeme přirozeně ztotožnit s jazykem $L_P \subseteq \Sigma^*$, který sestává z kódů pozitivních instancí problému P; rozumí se, že instance kódujeme řetězci v příslušné abecedě Σ. V doplňku L_P, tedy v množině $\Sigma^* \setminus L_P$, jsou pak kódy negativních instancí a také řetězce, které nejsou kódem instancí problému P. Např. L_{SAT} sestává ze zápisů splnitelných booleovských formulí, v jeho doplňku jsou pak zápisy nesplnitelných formulí a také řetězce v příslušné abecedě, které nejsou booleovskými formulemi.

O jazyku $L \subseteq \Sigma^*$ řekneme, že je rozhodnutelný, jestliže problém příslušnosti k L je rozhodnutelný, tedy jestliže existuje algoritmus, který pro každé $w \in \Sigma^*$ po konečném výpočtu zjistí, zda platí $w \in L$ či $w \notin L$.

Diskutovali jsme, že algoritmy přirozeně ztotožňujeme s programy v známých programovacích jazycích. Když jsme se pak zamysleli, co vlastně z hlediska vnějšího pozorovatele děláme, když provádíme výpočet daného programu na daném vstupu, dospěli jsme vcelku přímočaře k pojmu Turingův stroj, který jsme definovali jako strukturu

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F),$$

kde Q je konečná množina stavů, $q_0 \in Q$ je počáteční (neboli iniciální) stav, $F \subseteq Q$ je množina koncových stavů, Σ je konečná vstupní abeceda, Γ je konečná pásková abeceda, kde $\Sigma \subseteq \Gamma$ a pro speciální "prázdný znak" $\Box$ platí $\Box \in \Gamma \setminus \Sigma$, a "jádro stroje", tedy přechodová funkce δ, je typu

$$\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\}.$$

Pro exaktní vyjádření pojmu výpočet Turingova stroje jsme zavedli pojem *konfigurace stroje* M, což je řetězec z (regulárního) jazyka $\Gamma^* Q \Gamma^*$. Na množině konfigurací přirozeně zavedeme relaci $\vdash_M$ (odvození jedním krokem) touto definicí:

- když $\delta(q, a) = (q', b, 0)$, tak $uqav \vdash_M uq'bv$ pro každé $u, v \in \Gamma^*$;
- když $\delta(q, a) = (q', b, +1)$, tak $uqav \vdash_M ubq'v$ pro každé $u, v \in \Gamma^*$;
- když $\delta(q, a) = (q', b, -1)$, tak $uxqav \vdash_M uq'xbv$ pro každé $x \in \Gamma$ a $u, v \in \Gamma^*$.

Pro přesnost dodáme, že konfiguraci qv ztotožňujeme s konfigurací $\Box qv$ a uq ztotožňujeme s $uq\Box$ (a tedy konfiguraci q ztotožňujeme s $\Box q\Box$).

Iniciální (neboli počáteční) konfigurace pro (vstupní) slovo $w \in \Sigma^*$ je řetězec $q_0 w$.

Výpočet stroje $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ *na* vstupním *slově* $w \in \Sigma^*$ chápeme jako posloupnost konfigurací $C_0, C_1, C_2, \ldots$, kde C_0 je $q_0 w$ a pro $i = 0, 1, 2, \ldots$ máme $C_i \vdash_M C_{i+1}$ (tedy C_{i+1} dostaneme z C_i jedním krokem, příslušnou aplikací přechodové funkce δ). Výpočet může skončit v koncové konfiguraci, tj. v konfiguraci uqv, kde $q \in F$, nebo být nekonečný.

Při výše uvedené diskusi jsme mj. narazili na potřebnou proceduru "posun obsahu pásky". Navrhli jsme pro tuto proceduru konkrétní Turingův stroj, na němž jsme si zavedené pojmy ilustrovali.

Bavili jsme se o tom, že Turingovy stroje lze také chápat jako programy v "nejjednodušším univerzálním programovacím jazyku"; slovo *univerzální* zde odkazuje k možnosti "naprogramovat všechny algoritmy".

Zakončili jsme diskusí *Church-Turingovy teze*, kterou můžeme stručně vyjádřit takto:

$$\text{Algoritmus} = \text{Turingův stroj}.$$

Když později dokážeme, že nějaký problém je algoritmicky neřešitelný, tak to bude ve skutečnosti tak, že ukážeme, že není řešitelný žádným Turingovým strojem, a tiše se odvoláme na Church-Turingovu tezi.

Cvičení. Konstrukce jednoduchých Turingových strojů (které např. rozhodují (nebezkontextový) jazyk $\{wcw \mid w \in \{a, b\}^*\}$ či $\{ww \mid w \in \{a, b\}^*\}$, provádějí zdvojení slov, tedy w převedou na ww, apod.).

Týden 2

Pokračovali jsme v ilustraci myšlenek sestavování jednoduchých Turingových strojů.

Simulace mezi různými variantami Turingových strojů. Můžeme mj. odkázat na kapitolu 6.4 v [JAN-TI]. (Zájemce se samozřejmě může seznámit i s modelem RAM v části 6.3., v tomto kurzu se mu ale nebudeme věnovat.)

Připomněli jsme, že výpočet Turingova stroje $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ na vstupním slově $w \in \Sigma^*$ chápeme jako posloupnost konfigurací $C_0, C_1, C_2, \ldots$, kde C_0 je iniciální konfigurace $q_0 w$ a pro $i = 0, 1, 2, \ldots$ máme $C_i \vdash_M C_{i+1}$ (tedy C_{i+1} dostaneme z C_i jedním krokem, aplikací přechodové funkce δ). Výpočet může skončit v koncové konfiguraci (se stavem $q \in F$), nebo být nekonečný. Znakem $\vdash_M^*$ označujeme reflexivní a tranzitivní uzávěr relace $\vdash_M$, tedy

$$C \vdash_M^* C'$$

označuje, že výpočet stroje M se z konfigurace C dostane do C' konečným počtem kroků.

Řekneme, že *stroj* $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ *je simulován strojem* $M' = (Q', \Sigma', \Gamma', \delta', q_0', F')$, jestliže existuje jednoduché kódování konfigurací C stroje M konfiguracemi $\textsc{cod}(C)$ stroje M' (kde zobrazení COD a jeho inverze jsou realizovány jednoduchými algoritmy), tak, že ke každému výpočtu

$$C_0, C_1, C_2, \ldots, C_k \tag{1}$$

stroje M je výpočet stroje M' začínající v $C_0' = \textsc{cod}(C_0)$ tvaru

$$C_0', C_1', C_2', \ldots, C_{k'}', \tag{2}$$

kde existují $i_0 < i_1 < i_2 < \cdots < i_k$ pro něž platí $i_0 = 0$, $i_k = k'$ a $C_{i_j}' = \textsc{cod}(C_j)$ pro vš. $j \in [0, k]$. Každá konfigurace z výpočtu (1) má tedy svůj obraz ve výpočtu (2), ale jeden krok výpočtu (1) je simulován obecně konečnou sekvencí kroků ve výpočtu (2). Tuto definici lze přirozeně rozšířit i na nekonečné výpočty.

Jako příklad můžeme uvést simulaci Turingova stroje s oboustranně nekonečnou páskou strojem s jednostranně nekonečnou páskou. Konkrétně ke stroji $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ (při jehož výpočtu je možné, že pásková hlava se dostane i vlevo od vstupního slova) můžeme navrhnout stroj $M' = (Q', \Sigma, \Gamma', \delta', q_0', F)$ takto:

V prvé řadě předpokládáme, že stroj M nepoužívá k zápisu symbol $\Box$, tedy $\delta(q, a) = (q', a', m)$ implikuje $a' \neq \Box$. (Pokud by to nebylo splněno, tak prostě ke Γ přidáme "náhradní (zapisovací) prázdný symbol" $\Box'$ a funkci δ příslušně upravíme.)

Při konstrukci M' položíme $\Gamma' = \Gamma \cup \{¢\}$, kde $¢ \notin \Gamma$ je speciální symbol sloužící pro označení začátku pásky, a použijeme opakovaně proceduru "posun obsahu pásky" (o jedno políčko doprava), kterou jsme diskutovali minule. Konkrétně tedy Q' vytvoříme rozšířením Q o stavy

q_0',
q_x pro vš. $x \in \Sigma$,
q_L,
s_q pro vš. $q \in Q$,
$s_{(x,q)}$ pro vš. $q \in Q$ a $x \in \Gamma \setminus \{\Box\}$,
$s_{(L,q)}$ pro vš. $q \in Q$,

a funkce δ' pak vznikne rozšířením funkce δ takto:
(chceme, aby pro konfiguraci q'_0w, $w \in \Sigma^*$, platilo $q'_0w \vdash^*_{M'} \text{¢}q_0w$)
$\delta'(q'_0, \square) = (q_0, \text{¢}, +1)$
$\delta'(q'_0, x) = (q_x, \text{¢}, +1)$ pro vš. $x \in \Sigma$
$\delta'(q_x, y) = (q_y, x, +1)$ pro vš. $x \in \Sigma$, $y \in \Sigma$
$\delta'(q'_x, \square) = (q_L, x, -1)$ pro vš. $x \in \Sigma$
$\delta'(q_L, x) = (q_L, x, -1)$ pro vš. $x \in \Sigma$
$\delta'(q_L, \text{¢}) = (q_0, \text{¢}, +1)$
(pro konfiguraci $q\text{¢}v$, $q \in Q$ [hlava přišla na ¢ ve stavu q], chceme docílit $q\text{¢}v \vdash^*_{M'} \text{¢}q\square v$)
$\delta'(q, \text{¢}) = (s_q, \text{¢}, +1)$
$\delta'(s_q, \square) = (q, \square, 0)$
$\delta'(s_q, x) = (s_{(x,q)}, \square, +1)$ pro vš. $x \in \Gamma \setminus \square$
$\delta'(s_{(x,q)}, y) = (s_{(y,q)}, x, +1)$ pro vš. $x, y \in \Gamma \setminus \square$
$\delta'(s_{(x,q)}, \square) = (s_{(L,q)}, x, -1)$ pro vš. $x \in \Gamma \setminus \square$
$\delta'(s_{(L,q)}, x) = (s_{(L,q)}, x, -1)$ pro vš. $x \in \Gamma \setminus \square$
$\delta'(s_{(L,q)}, \square) = (q, \square, 0)$

Na přednášce jsme podrobně popsali, jak lze model dvouhlavých Turingových strojů simulovat standardním modelem (jednohlavých Turingových strojů). Ukázali jsme tedy jak k 2H-TS $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, v němž máme $\delta : (Q \setminus F) \times \Gamma \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\} \times \Gamma \times \{-1, 0, +1\}$ a jehož výpočet je příslušně definován, sestrojit 1H-TS $M' = (Q', \Sigma, \Gamma', \delta', q'_0, F')$ (v němž máme $\delta' : (Q' \setminus F') \times \Gamma' \to Q' \times \Gamma' \times \{-1, 0, +1\}$), který simuluje stroj M. (Myšlenka byla jednoduchá: M' si na pásce značí pozice obou hlav stroje M; jeden krok stroje M je pak simulován příslušnou sekvencí kroků stroje M', která realizuje zápis a posun u obou značek pro hlavy stroje M.)

Pak jsme se zamysleli nad simulací dvoupáskového stroje standardním jednopáskovým. Mj. jsme to udělali i technikou "dvoustopé pásky".

Cvičení. Simulace k-páskového stroje jednopáskovým a také stroje s dvourozměrnou páskou (hlava může chodit nejen doprava a doleva, ale také nahoru a dolů) standardním jednopáskovým.

Týden 3

(Cantorova) diagonalizační metoda. Připomněli jsme si tuto metodu nejprve na důkazu faktu, že jazyků nad dvouprvkovou abecedou $\Sigma = \{0, 1\}$ je nespočetně mnoho, tedy že množina

$$\mathcal{L} = \{L \mid L \subseteq \Sigma^*\}$$

je nekonečná množina, která není spočetná.

Představili jsme si nekonečnou (dvourozměrnou) tabulku, jejíž sloupce jsou označeny slovy $w_0, w_1, w_2, \ldots$ v dané abecedě Σ (těch je spočetně mnoho, neboť prvky $w \in \Sigma^*$ lze uspořádat např. podle délky a v rámci stejné délky lexikograficky, což je znázorněno zápisem $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$). Pro účely důkazu sporem jsme předpokládali, že jazyků $L \subseteq \Sigma^*$ je také spočetně mnoho a řádky tabulky jsme pak označili jazyky $L_0, L_1, L_2, \ldots$, přičemž jsme předpokládali, že každý jazyk $L \subseteq \Sigma^*$ se v tomto seznamu vyskytuje (tedy pro každý $L \subseteq \Sigma^*$ existuje $i \in \mathbb{N}$, pro něž $L = L_i$).

Do průsečíku řádku i a sloupce j jsme napsali 1, jestliže $w_j \in L_i$, a 0, jestliže $w_j \notin L_i$. Pak jsme vzali doplněk jazyka určeného diagonálou: definovali jsme tedy $L = \{w_i \mid w_i \notin L_i\}$ (podrobněji psáno $L = \{w \in \Sigma^* \mid \text{pro } i \text{ splňující } w = w_i \text{ platí } w_i \notin L_i\}$). Měli bychom podle předpokladu mít $L = L_i$ pro nějaké $i \in \mathbb{N}$; zkoumejme pro ono i, zda $w_i \in L$:

- když $w_i \in L$, tak podle definice jazyka L máme $w_i \notin L_i$, což je ve sporu s předpokladem $L = L_i$;
- když $w_i \notin L$, tak podle definice jazyka L máme $w_i \in L_i$, což je ve sporu s předpokladem $L = L_i$.

Tedy předpoklad, že L je v seznamu $L_0, L_1, L_2, \ldots$, vede ke sporu. Vyvodíme tedy, že nemůže existovat seznam $L_0, L_1, L_2, \ldots$, který obsahuje všechny prvky množiny $\mathcal{L}$; množina $\mathcal{L}$ tedy není spočetná.

Cvičení. Ukažte, že množina $\mathcal{P}(\mathbb{N}) = \{X \mid X \subseteq \mathbb{N}\}$ je nespočetná. Pak otázku spočetnosti/nespočetnosti zvažte pro množinu jazyků nad jednoprvkovou abecedou, tedy pro množinu $\{L \mid L \subseteq \{a\}^*\}$ pro jakýkoli pevně zvolený symbol a.

Nerozhodnutelnost problému zastavení (Halting problem, HP).
Uvažovali jsme rozhodovací (neboli ANO/NE) problém

> Halting Problem (HP)
>
> *Instance:* Turingův stroj M a slovo w v jeho vstupní abecedě.
>
> *Otázka:* Zastaví se M na w (neboli je výpočet stroje M pro w konečný)?

To nás přirozeně přivedlo ke *kódům Turingových strojů* (připomněli jsme si, že vstupní objekty problému potřebujeme nějak jednoduše kódovat řetězci ve vhodné abecedě, máme-li zvažovat algoritmickou řešitelnost). Nejdříve jsme se bez újmy na obecnosti omezili na Turingovy stroje se vstupní abecedou $\Sigma = \{0, 1\}$ a páskovou abecedou $\Gamma = \{0, 1, \square\}$ (prodiskutovali jsme způsob, jak lze libovolný Turingův stroj simulovat strojem s takto omezenými abecedami).

Načrtli jsme si konkrétní přirozený postup, jak lze Turingův stroj M přirozeně prezentovat kódem $\langle M \rangle$, což je řetězec v abecedě $\{0,1\}$. (Pro pohodlí můžeme samozřejmě nejprve použít trošku větší abecedu, ale je nám jasné, že každý symbol větší abecedy lze nahradit dohodnutým řetízkem bitů dostatečné délky.)

Jazyk kódů Turingových strojů, tedy

$$\{u \in \{0,1\}^* \mid u = \langle M \rangle \text{ pro nějaký stroj } M\},$$

je očividně rozhodnutelný a snadno navrhneme Turingův stroj, který pro zadané číslo $i \in \mathbb{N}$ vypíše kód stroje M_i, ze seznamu $M_0, M_1, M_2, \ldots$ všech strojů, které lze uspořádat podle délky kódu a v rámci stejné délky abecedně. (Takový stroj prostě generuje postupně všechny řetězce v $\{0,1\}^*$, o každém zjistí, zda je nebo není kódem nějakého Turingova stroje, a tím postupně dostává kódy $\langle M_0 \rangle, \langle M_1 \rangle, \langle M_2 \rangle, \ldots$ a nakonec vydá $\langle M_i \rangle$ pro zadané i.)

Výše uvedený diagonalizační postup jsme teď upravili takto:

Sloupce tabulky jsou stále označeny (všemi) slovy $w_0, w_1, w_2, \ldots$ v abecedě $\{0,1\}$, ale řádky teď označíme stroji $M_0, M_1, M_2, \ldots$, tedy všemi Turingovými stroji (s uvedenými abecedami $\{0,1\}$ a $\{0,1,\square\}$). (Turingových strojů je očividně jen spočetně mnoho, protože odpovídají svým kódům, tedy spočetně mnoha slovům v abecedě $\{0,1\}$.)

Pro účely sporu jsme předpokládali, že existuje stroj T_{HP}, který rozhoduje problém HP, a navrhli jsme konkrétní stroj D, který pracuje takto:

> Pro zadané $w \in \{0,1\}^*$ zjistí jeho pořadí i (tedy $w = w_i$ v seznamu označujícím sloupce), zkonstruuje stroj M_i (což lze, jak jsme diskutovali výše) a pomocí T_{HP} zjistí, zda M_i se zastaví na w_i; když ano, tak D skočí do nekonečného cyklu, když ne, tak se D zastaví (tedy výpočet ukončí skokem do koncového stavu).

Jelikož D musí být v seznamu $M_0, M_1, M_2, \ldots$, máme $D = M_i$ pro nějaké konkrétní $i \in \mathbb{N}$. Uvažme, zda D se zastaví na vstup w_i:

- když se D zastaví na w_i, tak se M_i zastaví na w_i (protože $D = M_i$), ale v tom případě se D na w_i nezastaví - spor;
- když se D nezastaví na w_i, tak se M_i nezastaví na w_i (protože $D = M_i$), ale v tom případě se D na w_i zastaví - spor.

Dosažený spor nutně znamená, že předpoklad, že existuje příslušný T_{HP}, je nepravdivý. Problém HP tedy není rozhodován žádným Turingovým strojem; díky přijímání Church-Turingovy teze říkáme, že HP je (algoritmicky) nerozhodnutelný.

Cvičení. Mj. důkaz nerozhodnutelnosti HP přes problém označený jako *diagonální problém zastavení* (DHP) v části 6.5 [JAN-TI].

Týden 4

Částečně rozhodnutelné problémy a jazyky (rozšíření rozhodnutelných).
Definovali jsme *jazyky přijímané Turingovými stroji*, také se jim říká *rekurzivně spočetné* nebo *částečně rozhodnutelné*.

Konkrétně: Pro Turingův stroj $M = (Q, \Sigma, \Gamma, \delta, q_0, \{q_+, q_-\})$ definujeme jazyk

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash_M^* uq_+v \text{ pro nějaké } u, v \in \Gamma^*\};$$

o jazyku $L(M)$ říkáme, že je přijímán (nebo též částečně rozhodován) strojem M.

Jazyk $L \subseteq \Sigma^*$ je *rekurzivně spočetný*, jestliže je přijímán nějakým Turingovým strojem, tedy jestliže $L = L(M)$ pro nějaký stroj M. Třídu rekurzivně spočetných jazyků označujeme RecEnum (recursively enumerable).

Podtřídou RecEnum je třída Rec obsahující *jazyky rozhodované Turingovými stroji*, kterým se také říká *rekurzivní* nebo *rozhodnutelné*. O výše definovaném jazyku $L(M)$ řekneme, že je (nejen přijímán, ale i) *rozhodován* strojem M, jestliže navíc pro každé $w \in \Sigma^* \setminus L(M)$ platí $q_0 w \vdash_M^* uq_-v$ pro nějaké $u, v \in \Gamma^*$.

Třídy těchto jazyků (Rec a RecEnum) jsou pochopitelně nekonečné, ale spočetné, protože každý Turingův stroj M můžeme zadat jeho kódem $\langle M \rangle$, tedy slovem v pevně dané abecedě (vystačíme si samozřejmě i s posloupností bitů, tedy slovem v abecedě $\{0, 1\}$); víme, že slov v pevně zvolené abecedě je (jen) spočetně mnoho.

O ANO/NE problému řekneme, že je *částečně rozhodnutelný*, jestliže jazyk sestávající z (kódů) jeho pozitivních instancí je přijímán nějakým Turingovým strojem, tedy (při připomenutí Church-Turingovy teze) jestliže existuje algoritmus, jehož běh (výpočet) pro každou pozitivní instanci problému skončí s odpovědí ANO a pro každou negativní instanci je buď nekonečný nebo skončí s odpovědí NE. (Někdy se v definici pro jednoduchost uvádí, že výpočty pro negativní instance jsou nekonečné. Uvědomili jsme si, že pojem částečně rozhodnutelného problému se tím nezmění.)

Univerzální Turingův stroj.
Uvědomili jsme si, že dokážeme vcelku snadno sestrojit (lépe řečeno popsat, jak lze sestrojit) *univerzální Turingův stroj*:

Věta 1 *Existuje Turingův stroj U, který pro vstupní slovo tvaru $\langle M \rangle w$ simuluje výpočet stroje M na w.*

Důkaz. (Idea.) U nejdříve zkontroluje, zda vstupní slovo je tvaru $\langle M \rangle w$ pro nějaký stroj M a nějaké slovo $w \in \{0, 1\}^*$, a v kladném případě pak simuluje M na slově w. Omezili jsme se na stroje M, které mají vstupní abecedu $\{0, 1\}$ a páskovou abecedu $\{0, 1, \square\}$, ale to není žádné reálné omezení, jak víme. Také jsme si všimli, že U se samozřejmě navrhuje snadněji, chápeme-li ho jako vícepáskový a mající bohatší páskovou abecedu; na jednopáskový stroj s páskovou abecedou $\{0, 1, \square\}$ se pak dá rutinně převést, jak již víme. $\square$

Halting problem je částečně rozhodnutelný. O jazyku

$$L_{HP} = \{\langle M \rangle w \mid \text{výpočet stroje } M \text{ na } w \text{ se zastaví}\}$$

již víme, že je nerozhodnutelný; univerzální Turingův stroj prokazuje, že je ale částečně rozhodnutelný. (Proč?) Jinými slovy: Halting Problem (HP) je příklad problému, který je částečně rozhodnutelný, ale ne rozhodnutelný.

Postova věta.
Uvědomili jsme si triviální fakt, že třída Rec je uzavřena na doplněk:

Tvrzení 2 *Jazyk $L \subseteq \Sigma^*$ je rozhodnutelný právě tehdy, když jeho doplněk $\overline{L} = \Sigma^* \setminus L$ je rozhodnutelný.*

Ovšem třída RecEnum na doplněk uzavřena není. To plyne z následující věty (nazývané Postova věta):

Věta 3 *Jazyk L je rozhodnutelný právě tehdy, když L i $\overline{L}$ (tj. doplněk jazyka L) jsou částečně rozhodnutelné.*

> Připomeňme si, že přesnější by bylo větu formulovat jako "L je rekurzivní právě tehdy, když L i $\overline{L}$ jsou rekurzivně spočetné". Pojmy "rekurzivní" a "rekurzivně spočetný" odkazují k Turingovým strojům, zatímco "rozhodnutelný" a "částečně rozhodnutelný" k algoritmům. My jsme ovšem diskutovali tzv. *Church-Turingovu tezi* (viz také 6.4 v [JANTI]), díky níž příslušné pojmy (jako "rekurzivní" a "rozhodnutelný") ztotožňujeme.

Důkaz. Jelikož rozhodnutelný jazyk je i částečně rozhodnutelný (proč?), tak jeden směr tvrzení věty plyne hned z Tvrzení 2.

Pro druhý směr předpokládáme, že L i $\overline{L}$ jsou částečně rozhodnutelné; existuje tedy algoritmus (Turingův stroj) A_1, který přijímá (neboli částečně rozhoduje) jazyk L_1, a algoritmus (Turingův stroj) A_2, který přijímá (neboli částečně rozhoduje) jazyk L_2. Navrhneme algoritmus (Turingův stroj) A, který rozhoduje jazyk L: na vstupu w algoritmus A souběžně simuluje algoritmy A_1 i A_2; je zřejmé, že právě jeden z nich skončí s odpovědí ANO. Pokud s odpovědí ANO skončí A_1, algoritmus A vydá ANO ($w \in L$), a pokud s odpovědí ANO skončí A_2, algoritmus A vydá NE ($w \notin L$). □

K ANO/NE problému P přirozeně zavedeme pojem *doplňkového problému* co-P (odpovědi ANO a NE jsou prohozeny) a všimneme si jednoduchého důsledku našich dosavadních zjištění:

Tvrzení 4 *Problém HP je částečně rozhodnutelný, ale není rozhodnutelný. Problém co-HP není (ani) částečně rozhodnutelný.*

(Algoritmická) převeditelnost mezi problémy (mapping reducibility $\leq_{\mathrm{m}}$).
Bavili jsme se o problému UHP (Universal Halting Problem), kde instancí je Turingův stroj M a otázkou je, zda se M zastaví na každý svůj vstup. Přišli jsme na tento vztah:

Tvrzení 5 *Kdyby UHP byl rozhodnutelný, tak by i HP byl rozhodnutelný.*

Z toho plyne, že UHP rozhodnutelný není.

Uvědomili jsme si následující zobecnění našeho postupu. Začali jsme definicí.

Pro jazyky $L_1, L_2 \subseteq \Sigma^*$ platí

$$L_1 \leq_{\mathrm{m}} L_2,$$

což čteme L_1 je *převeditelný*, nebo též *redukovatelný*, *na* L_2, jestliže existuje vyčíslitelná funkce $f : \Sigma^* \to \Sigma^*$ taková, že $\forall w \in \Sigma^* : w \in L_1 \Leftrightarrow f(w) \in L_2$.

Připomněli jsme si, co znamená pojem "vyčíslitelná funkce", a uvedli si význam indexu m v $\leq_{\mathrm{m}}$. (Jedná se totiž o tzv. "mapping reducibility".) Také víme, že každému ANO/NE-problému P odpovídá jazyk L_P (množina všech zápisů instancí problému P s odpovědí ANO). Proto u ANO/NE-problémů P_1, P_2 budeme také používat zápis $P_1 \leq_{\mathrm{m}} P_2$ (čteme: problém P_1 je redukovatelný na problém P_2); tento zápis můžeme formálně chápat jako zkratku za $L_{P_1} \leq_{\mathrm{m}} L_{P_2}$. Tedy

$$P_1 \leq_{\mathrm{m}} P_2$$

platí právě tehdy, když existuje algoritmus, který ke každé instanci I_1 problému P_1 vydá instanci I_2 problému P_2 tak, že jsou zaručeny tyto dvě podmínky:

1. když I_1 je pozitivní instancí problému P_1, tak I_2 je pozitivní instancí problému P_2;
2. když I_1 je negativní instancí problému P_1, tak I_2 je negativní instancí problému P_2.

Pak jsme jednoduše dokázali několik důležitých faktů:

Tvrzení 6 *Nechť* $L_1 \leq_{\mathrm{m}} L_2$.
a) Jestliže L_2 *je rozhodnutelný, pak* L_1 *je rozhodnutelný.*
(Neboli: jestliže L_1 *je nerozhodnutelný, pak* L_2 *je nerozhodnutelný.)*
b) Jestliže L_2 *je částečně rozhodnutelný, pak* L_1 *je částečně rozhodnutelný.*
(Neboli: jestliže L_1 *není částečně rozhodnutelný, pak* L_2 *není částečně rozhodnutelný.)*

Uvědomili jsme si, že podstatou našeho důkazu Tvrzení 5 bylo prokázání, že HP$\leq_{\mathrm{m}}$UHP. (Navrhli jsme algoritmus, který pro vstup M, w vydá výstup M' tak, že M se zastaví na w právě tehdy, když M' se zastaví na všechny vstupy.)

Cvičení. Další ilustrace nově zavedených pojmů a tvrzení.

Týden 5

Ani jeden z problémů UHP a co-UHP není částečně rozhodnutelný.
Připomeňme, že jsme si ukázali, že platí HP$\leq_{\mathrm{m}}$UHP (a tedy UHP není rozhodnutelný). Všimněme si:

Pozorování 7 *$P \leq_{\mathrm{m}} P' \Leftrightarrow$ co-$P \leq_{\mathrm{m}}$ co-P'.*

Tedy máme co-HP$\leq_{\mathrm{m}}$co-UHP, z čehož plyne, že doplňkový problém problému UHP (zformulujte si jej) není částečně rozhodnutelný. Pak jsme si ukázali následující tvrzení, z něhož plyne, že problém UHP také není částečně rozhodnutelný.

Tvrzení 8 *co-HP$\leq_{\mathrm{m}}$ UHP.*

Důkaz. Problém co-HP je doplňkový (complementary) problém k problému zastavení; instance je tedy M, w a otázkou je, zda je výpočet M nad w nekonečný. Máme tedy navrhnout algoritmus, který k zadanému M, w sestrojí stroj M' tak, že je zaručeno, že

- když výpočet M na w je nekonečný, tak výpočty M' jsou na všech vstupech konečné;
- když výpočet M na w je konečný, tak existuje vstup u stroje M' takový, že výpočet M' na u je nekonečný.

Nejdříve si připomeňme, že výpočty Turingových strojů lze přirozeně popisovat řetězci ve vhodné abecedě Δ; výpočet (nebo též "výpočetní historie") je prostě příslušná posloupnost konfigurací oddělených speciálním znakem: např. $\#C_0\#C_1\#C_2\#\cdots C_k\#$. Všimneme si teď triviálního faktu:

- když výpočet M na w je konečný,
  tak existuje (konečný) řetězec $u \in \Delta^*$ ve tvaru $\#C_0\#C_1\#C_2\#\cdots C_k\#$, kde C_0 je počáteční konfigurace q_0w, C_k je koncová konfigurace (stav v ní je koncový) a platí $C_i \vdash_M C_{i+1}$ pro vš. $i = 0, 1, \ldots, k{-}1$ (tedy C_{i+1} vznikne z C_i jedním krokem stroje M);
- když výpočet M na w je nekonečný,
  tak pro každý řetězec $u \in \Delta^*$ platí, že je v něm "chyba", což znamená, že platí jedna z těchto možností:
  a) u není ve tvaru $\#C_0\#C_1\#C_2\#\cdots C_k\#$, kde všechny řetězce C_i jsou konfiguracemi stroje M, nebo
  b) u je ve tvaru $\#C_0\#C_1\#C_2\#\cdots C_k\#$, kde všechny řetězce C_i jsou konfiguracemi stroje M, ale v tom případě nastává alespoň jedna z těchto možností:
     i) C_0 není konfigurace q_0w,
     ii) C_k není koncová konfigurace,
     iii) pro nějaké $i \in [0, k{-}1]$ neplatí $C_i \vdash_M C_{i+1}$.

Stačí tedy navrhnout M', který pro vstupní $u \in \Delta^*$ systematicky hledá výše uvedenou chybu; když ji najde, zastaví se, ale když tam chyba není (tedy když u popisuje kompletní konečný výpočet M na w), tak M' skočí do nekonečného cyklu (nezastaví se). □

Již jsme věděli, že co-HP nepatří do RecEnum; jeho doplňkový problém, tedy HP, do RecEnum ovšem patří. Problém UHP je tedy "ještě těžší" než co-HP: není v RecEnum a ani jeho doplňkový problém není v RecEnum. Studují se celé hierarchie problémů ležících mimo RecEnum, např. tzv. aritmetická hierarchie, ale to je mimo záběr našeho kurzu.

Problém ekvivalence bezkontextových gramatik je nerozhodnutelný.
Diskutovali jsme o tom, že problémy prázdnosti E_{REG} a E_{CFL} pro regulární, resp. bezkontextové, jazyky jsou rozhodnutelné; i problém úplnosti (nebo též univerzality) All_{REG} je rozhodnutelný.

Pak jsme ukázali:

Tvrzení 9 *Problém All_{CFL} (Instance: bezkontextová gramatika $G = (\Pi, \Sigma, S, P)$. Otázka: $L(G) = \Sigma^*$?) není rozhodnutelný, dokonce ani částečně rozhodnutelný.*

Důkaz. Vzpomněli jsme si na důkaz Tvrzení 8 a ukázali jsme, že platí co-HP$\leq_{\mathrm{m}}$ All_{CFL}. Klíčovým faktem je, že odhalení chyby v $u \in \Delta^*$, tedy ověření, že u není ve tvaru $\#C_0\#C_1\#C_2\#C_3\#\cdots C_k\#$, který popisuje kompletní výpočet stroje $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ na w, lze provést nedeterministickým zásobníkovým automatem (NZA). Mnohé chyby v popisu (C_0 není tvaru q_0w, nějaký řetězec mezi dvěma symboly # neobsahuje symbol $q \in Q$ nebo jich obsahuje víc, poslední řetězec neobsahuje $q \in F$) lze ověřit konečným automatem (který lze chápat jakou součást řídicí jednotky NZA). Nad ověřením $C_i \not\vdash_M C_{i+1}$ je ovšem potřeba se trochu zamyslet. (Dále si pak stačí připomenout, že k nedeterministickému zásobníkovému automatu lze snadno sestrojit ekvivalentní bezkontextovou gramatiku.)

Podívejme se tedy na to, jak může NZA ověřit, že řetězec $\#C\#C'\#$ nesplňuje $C \vdash_M C'$; nechť $C = a_1a_2\cdots a_n$ a $C' = a'_1a'_2\cdots a'_{n'}$. Pokud a_1a_2 není tvaru qa, kde $\delta(q,a) = (.,.,-1)$ (posun hlavy doleva), tak lze $C \not\vdash_M C'$ prokázat porovnáním určité trojice $a_ia_{i+1}a_{i+2}$ s trojicí $a'_ia'_{i+1}a'_{i+2}$ (proč?), případně zjištěním, že $n > n'$. Jelikož NZA může uhodnout i, které si odpočítá pomocí zásobníku, může toto porovnání provést. V případě $a_1a_2 = qa$, kde $\delta(q,a) = (q', a', -1)$, musí řetězec C' začínat $a'_1a'_2a'_3 = q'\Box a'$ (jinak je už jasné, že $C \not\vdash_M C'$) a v tom případě se porovnají příslušné trojice $a_ia_{i+1}a_{i+2}$ a $a'_{i+1}a'_{i+2}a'_{i+3}$. (Domyslete i případ, kdy je chyba až na konci a C končí symbolem $q \in Q$.) □

Na cvičení jsme si mj. ukázali, že $All_{CFL} \leq_{\mathrm{m}} EQ_{CFL}$; problém EQ_{CFL} je definován takto: Instance: bezkontextové gramatiky G_1, G_2; Otázka: $L(G_1) = L(G_2)$?

Pro problém EQ_{CFL} tedy dostáváme, že není (ani) částečně rozhodnutelný. Na druhé straně, jeho doplňkový problém $NonEQ_{CFL}$ částečně rozhodnutelný je (generujeme systematicky všechna terminální slova a ověřujeme pro každé z nich, zda jej generuje právě jedna z daných dvou gramatik).

Riceova věta.
Uvědomili jsme si, že každá vlastnost V Turingových strojů (např. „stroj M má více než 100 stavů“, „výpočet stroje M je pro každý vstup konečný“, apod.) rozdělí množinu všech Turingových strojů na dvě disjunktní podmnožiny: jedna je množina strojů, které vlastnost V mají, a druhá je množina strojů, které vlastnost V nemají. Vlastnost V je *triviální*, jestliže je jedna z oněch dvou příslušných množin prázdná (tedy buď všechny stroje vlastnost V mají nebo ji nemá ani jeden). Vlastnost V je *netriviální*, jestliže ji alespoň jeden stroj má a alespoň jeden stroj nemá.

Důkladně jsme si promysleli, co to je *vstupně/výstupní vlastnost*, zkráceně též *I/O vlastnost*, Turingových strojů (či obecně „programů“).

Speciálně jsme si uvědomili, že vlastnost V není I/O vlastností právě tehdy, když existují dva stroje M_1, M_2, které mají stejnou I/O tabulku (tedy stejné vstupně/výstupní chování; v tabulce jsou v jednom sloupci všechny vstupy a v druhém příslušné výstupy včetně symbolu $\perp$ v případech, kdy je výpočet pro daný vstup nekonečný), ale jeden z nich vlastnost V má a druhý ji nemá.

Probrali jsme si vlastnosti podobné níže uvedeným (které jsou v řešeném příkladu 7.1. v kapitole 7 [JAN-TI]) a uvědomili si, které jsou I/O vlastnostmi. Speciálně jsme si všimli, že podle definice je každá triviální vlastnost I/O vlastností.

a/ Zastaví se M na řetězec 001 ?

b/ Má M více než sto stavů ?

c/ Má v nějakém případě výpočet stroje M více kroků než tisícinásobek délky vstupu ?

d/ Platí, že pro libovolné n se M na vstupech délky nejvýše n vícekrát zastaví než nezastaví ?

e/ Zastaví se M na každém vstupu w za méně než $|w|^2$ kroků?

f/ Je pravda, že pro lib. vstupní slovo M realizuje jeho zdvojení ?

g/ Je pravda, že M má nejvýše sto stavů nebo více než sto stavů ?

Uvědomme si, že každé vlastnosti V přirozeně odpovídá ANO/NE problém P_V, u nějž instancí je Turingův stroj M a otázkou je, zda M má vlastnost V. Z jiného pohledu: každé vlastnosti V odpovídá jazyk $L_V = \{\langle M\rangle \mid M \text{ je Turingův stroj, který má vlastnost } V\}$.

Pak jsme se zamysleli nad Riceovou větou:

> *Každá netriviální vstupně/výstupní vlastnost programů (tj. Turingových strojů) je nerozhodnutelná.*

Jinak řečeno: pro každou netriviální vstupně/výstupní vlastnost V je problém P_V nerozhodnutelný (neboli jazyk L_V nepatří do množiny Rec).

Důkaz Riceovy věty.
Uvažujme libovolnou netriviální vstupně/výstupní vlastnost V Turingových strojů. Ukážeme, že pro problém P_V (*Instance:* Turingův stroj M; *Otázka:* má M vlastnost V?) platí $\mathrm{HP} \leq_\mathrm{m} P_V$ nebo co-$\mathrm{HP} \leq_\mathrm{m} P_V$ (kde HP je Halting Problem). Tím bude důkaz Riceovy věty hotov. (Proč?)

Uvažujme teď jistý stroj $M_\perp$, který se nezastaví na žádný vstup (např. vždy hned skočí do nekonečného cyklu). Rozlišme dva případy:

a) stroj $M_\perp$ vlastnost V nemá;

b) stroj $M_\perp$ vlastnost V má.

Případ a)

Zvolme nějaký stroj M_1, který vlastnost V má. (Takový musí existovat, protože vlastnost V je netriviální.) Ukážeme, že platí HP$\leq_{\mathrm{m}} P_V$: uvážíme algoritmus, který k instanci HP, tedy ke stroji M a slovu w, sestrojí stroj M', popsaný takto:

> - Můžeme si představit, že M' je dvoupáskový (víme, že takový stroj lze simulovat standardním jednopáskovým strojem) a svůj vstup u dostane zapsaný na první pásce.
>
> - M' má ve svém počátečním stavu uloženo slovo w, které na začátku svého výpočtu napíše na druhou pásku a pro něj simuluje výpočet M (na druhé pásce, nezávisle na svém vstupu u zapsaném na první pásce).
>
> - Pokud simulace M na w skončí (tedy M se zastaví na w, a tedy M, w je pozitivní instancí problému HP), tak M' pokračuje simulací stroje M_1 na vstupu u na první pásce.

Vidíme, že platí:

- Pokud M, w je pozitivní instancí problému HP (M se zastaví na w), tak M' má stejné I/O chování (má stejnou I/O tabulku) jako M_1; v tom případě je M' pozitivní instancí problému P_V (M' musí mít vlastnost V, protože V je vstupně/výstupní vlastnost, M_1 má vlastnost V a M' má stejnou I/O tabulku jako M_1).
- Pokud M, w je negativní instancí problému HP (výpočet M na w je nekonečný), tak M' má stejné I/O chování (má stejnou I/O tabulku) jako $M_\perp$; v tom případě je M' negativní instancí problému P_V (protože $M_\perp$ nemá vlastnost V).

Případ b)

Zde stroj $M_\perp$ má vlastnost V; zvolíme stroj M_1, který vlastnost V nemá. Uvážíme algoritmus, který k M, w sestrojí M' úplně stejně jako výše. Ověřte, že v nyní uvažovaném případě tento algoritmus prokazuje, že co-HP$\leq_{\mathrm{m}} P_V$. □

Cvičení. Mj. jsme si ukázali jednoduchou redukci $All_{CFL} \leq_{\mathrm{m}} EQ_{CFL}$.
Také jsme se zamysleli nad otázkou, na které z následujících dvou vlastností se vztahuje Riceova věta a proč:

- Rozhoduje M ekvivalenci konečných automatů?
- Rozhoduje M ekvivalenci bezkontextových gramatik?