Vyčíslitelnost a složitost

(bc. předmět KMI/VS)

Petr Jančar

18. prosince 2017

Poznámky k textu. Tento text vzniká v průběhu kursu v zimním semestru 2017/18 jako podpůrný materiál k přednáškám a cvičením. (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 kursu 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. 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 předchozích běhů kursu dr. J. Konečného. Budeme také čerpat z pracovního materiálu autora textu používaného na VŠB-TU:
Jančar P.: Teoretická informatika (VŠB-TU Ostrava, 2010)
`www.cs.vsb.cz/jancar/TEORET-INF/ti-text.2010-01-20.pdf`.
(Tento text je také přímo přístupný na web-stránce předmětu.)

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).

Vedle knihy "Sipser M. Introduction to the Theory of Computation" zmíněné na úvodní stránce lze pro teorii vyčíslitelnosti 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).

Avizoval jsem první zápočtovou písemku v 6. týdnu semestru a druhou zápočtovou písemku v 12. týdnu (s možností oprav v posledním, 13. týdnu). První písemku může posluchač(ka) splnit tím, že v 5. týdnu vysvětlí na přednášce, jak je možné vyřešit následující úkol:

> Představte si, že chcete navrhnout rozšíření nějakého standardního programovacího jazyka tak, že v libovolném programu povolíte instrukci
>
> do proměnné *OwnCode* přiřaď tento program.
>
> Zamýšlený význam instrukce je ten, že když takovou instrukci použijeme v programu P, tak po jejím provedení bude v proměnné *OwnCode* uložen program P (jakožto řetězec znaků v jisté abecedě, prostě zdrojový kód programu P). S obsahem proměnné *OwnCode* lze pak samozřejmě dále pracovat, např. jej vytisknout.
>
> Lze vůbec instrukci s takovým zamýšleným významem nějak implementovat? Jinými slovy: lze program P s takovou instrukcí považovat za zkratku mechanicky zkonstruovatelného programu P-STAND, který obsahuje jen standardní instrukce a chová se, jak zamýšleno, tedy speciálně do proměnné *OwnCode* dostane svůj vlastní kód (tj. kód P-STAND)?
>
> Asi nás napadne implementace typu: otevři soubor P-STAND.ZDROJ a obsah souboru načti do proměnné *OwnCode*. Přitom ovšem musíme mít zaručeno, že v souboru P-STAND.ZDROJ je opravdu uložen zdrojový kód programu P-STAND. Když tam bude něco jiného, tak se program chová jinak, než zamýšleno; kromě předpokládaného vstupu, na němž má pracovat, jsme program P-STAND udělali závislým na ještě jiném vstupu (konkrétně na obsahu souboru P-STAND.ZDROJ).
>
> Kupodivu to jde šikovně zařídit i tak, aby příslušný P-STAND nebyl závislý na dalším vstupu, a vy právě toto máte vysvětlit.

Zájemce o tuto formu zápočtové písemky se musí přihlásit alespoň týden předem (poté, co už to má rozmyšleno); v případě více zájemců budeme situaci nějak rozumně řešit. (Není to hromadný úkol.) Poznamenejme ještě, že nejde o nějakou hříčku, ale o jeden z fundamentálních faktů teorie vyčíslitelnosti (jak uvidíme později).

Turingovy stroje.

Bavili jsme se o pojmu *algoritmus* a pak o *Turingových strojích* jakožto o "nejjednodušším univerzálním programovacím jazyku"; slovo *univerzální* zde odkazuje k možnosti "naprogramovat všechny algoritmy".

Obsah přednášky a cvičení je v zásadě obsažen v částech 6.1. a 6.2. textu [JAN-TI] (tak budeme dále odkazovat na text Jančar: Teoretická informatika, který je uveden na první straně a jenž je přímo přístupný z web-stránky předmětu).

Týden 2

Pokračovali jsme v ilustraci myšlenek sestavování jednoduchých Turingových strojů.

Speciálně jsme se soustředili na vzájemnou simulaci 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.)

Podrobněji jsme probrali simulaci Turingova stroje s oboustranně nekonečnou páskou strojem s jednostranně nekonečnou páskou. Udělali jsme to tentokrát technikou "dvoustopé pásky".

Připomněli jsme si Cantorovu diagonalizační metodu, konkrétně na důkazu faktu, že jazyků nad (již) dvouprvkovou abecedou je nespočetně mnoho:

Představili jsme si nekonečnou tabulku, jejíž sloupce jsou označeny slovy $w_0, w_1, w_2, \ldots$ v dané abecedě Σ (těch je spočetně mnoho, uspořádali jsme je podle délky a v rámci stejné délky lexikograficky). Pro účely důkazu sporem jsme předpokládali, že jazyků $L \subseteq \Sigma^*$ je také spočetně mnoho a řádky jsme pak označili jazyky $L_0, L_1, L_2, \ldots$, předpokládajíce, že každý jazyk $L \subseteq \Sigma^*$ se v tomto seznamu vyskytuje.

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 $L = \{w_i \mid w_i \notin L_i\}$. Měli bychom podle předpokladu mít $L = L_k$ pro nějaké $k \in \mathbb{N}$. Pak ovšem fakt $w_k \in L \Leftrightarrow w_k \notin L_k$ vede ke sporu.

Definovali jsme *jazyky rozpoznávané Turingovými stroji*, také se jim říká *rekurzivně spočetné* nebo *částečně rozhodnutelné*.

Jejich podtřídou jsou *jazyky rozhodované Turingovými stroji*, kterým se také říká *rekurzivní* nebo *rozhodnutelné*.

Třídy těchto jazyků jsou pochopitelně nekonečné, ale spočetné, protože každý Turingův stroj můžeme zadat slovem v pevně dané abecedě (vystačíme si samozřejmě i s posloupností bitů, tedy slovem v abecedě $\{0, 1\}$).

To nás přivedlo ke *kódům Turingových strojů*. Ukázali jsme si konkrétní přirozený postup, jak lze Turingův stroj M přirozeně prezentovat řetězcem $\langle M \rangle$ v abecedě $\{0, 1\}$. (Pro pohodlí jsme použili trošku větší abecedu, ale bylo jasné, že každý symbol lze nahradit dohodnutým řetízkem bitů.)

Uvědomili jsme si, že dokážeme vcelku snadno sestrojit (lépe řečeno popsat, jak lze sestrojit) *univerzální Turingův stroj* U. Ten nejdříve zkontroluje, zda slovo zadané na pásce 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 vícepaskový; na jednopáskový se pak dá převést standardně.)

Speciálně jsme si uvědomili, že U můžeme spustit i na svůj vlastní kód (tedy na vstup tvaru $\langle U \rangle w$).

Jazyk $\{u \in \{0, 1\}^* \mid u = \langle M \rangle$ pro nějaký stroj $M\}$ je očividně rozhodnutelný a existuje tedy Turingův stroj, který pro zadané číslo i vypíše kód stroje M_i, ze seznamu $M_0, M_1, M_2, \ldots$ všech strojů (zase lze uspořádat podle délky kódu a v rámci stejné délky abecedně).

Upravili jsme náš diagonalizační postup pro důkaz toho, že *problém zastavení* (halting problem, HP) není rozhodnutelný; jinými slovy, jazyk

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

je nerozhodnutelný:

Tentokrát jsme řádky označili stroji $M_0, M_1, M_2, \ldots$. Pro účely sporu jsme předpokládali, že existuje stroj T_{HP}, který rozhoduje L_{HP}, a navrhli jsme stroj M, který pracuje takto:

Pro zadané w 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 M skočí do nekonečného cyklu, když ne, tak se M zastaví.

Jelikož M musí být v seznamu, máme $M = M_k$ pro nějaké k. Má ovšem platit, že M se zastaví na w_k právě tehdy, když M_k se nezastaví na w_k — spor; tedy předpoklad, že existuje příslušný T_{HP}, je nepravdivý.

Jelikož U rozpoznává (také se říká *přijímá* nebo též *částečně rozhoduje*) jazyk L_{HP}, máme příklad jazyka, který je částečně rozhodnutelný, ale není rozhodnutelný.

Na cvičení jsme mj. dali dohromady důkaz nerozhodnutelnosti HP přes problém označený jako *diagonální problém zastavení* (DHP) v části 6.5 [JAN-TI].

Rovněž jsme diskutovali *nedeterministické Turingovy stroje* a uvědomili jsme si, že strom výpočtů nedeterministického stroje M_1 na slově w je možné konstruovat do šířky deterministickým strojem M_2. Jazyk přijímaný nedeterministickým strojem M (zde $w \in L(M)$ právě tehdy, když alespoň jeden výpočet na w je akceptující) je tedy také přijímán nějakým deterministickým strojem M'.

Můžeme také zavést pojem *rozhodování jazyka nedeterministickým strojem*; v tom případě požadujeme, ať jsou všechny výpočty konečné (akceptující nebo zamítající). Je vidět, že třída jazyků rozhodnutelných nedeterministickými stroji je totožná s třídou rozhodnutelných jazyků.

Týden 3

Připomněli jsme si pojem (formálních) gramatik. *Gramatika* je pro nás struktura $G = (\Pi, \Sigma, S, P)$, kde Π je konečná množina neterminálů, Σ konečná množina terminálů, $S \in \Pi$ je počáteční (neboli startovací) neterminál a P je konečná množina pravidel typu $\alpha \to \beta$, kde $\alpha \in (\Pi \cup \Sigma)^*\Pi(\Pi \cup \Sigma)^*$ a $\beta \in (\Pi \cup \Sigma)^*$. Na množině $(\Pi \cup \Sigma)^*$ je definována relace $\Rightarrow$ jako nejmenší relace splňující, že pro každé pravidlo $\alpha \to \beta$ v P platí $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ pro všechna $\gamma, \delta \in (\Pi \cup \Sigma)^*$. Relace $\Rightarrow^*$ je reflexivním a tranzitivním uzávěrem relace $\Rightarrow$. Gramatika G generuje jazyk $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$. Jazyky generované uvedenými gramatikami (kterým se také říká gramatiky typu 0, nebo neomezené gramatiky) se nazývají *jazyky typu* 0 v *Chomského hierarchii.*

Připomněli jsme, že jazyk $L \subseteq \Sigma^*$ je *rozpoznávaný* (nebo také *přijímaný*) Turingovým strojem M, jestliže pro každé $w \in \Sigma^*$ platí, že výpočet M na w je akceptující (neboli přijímající) právě tehdy, když $w \in L$ (tedy pro $w \in \Sigma^* \setminus L$ je výpočet buď zamítající nebo nekonečný).

Dokázali jsme následující tvrzení.

Tvrzení 1 *Třída jazyků rozpoznávaných Turingovými stroji je totožná s třídou jazyků typu* 0 *v Chomského hierarchii.*

Připomněli jsme si další stupně Chomského hierarchie:

Jazyky typu 1, nebo též kontextové jazyky (anglicky context-sensitive languages), jsou generovány gramatikami typu 1, tzv. nezkracujícími gramatikami, tj. gramatikami, v nichž pravidla $\alpha \to \beta$ musí splňovat podmínku $|\alpha| \leq |\beta|$ (tedy pravá strana pravidla není kratší než levá strana). Speciálně jsme si ukázali, že jazyk $\{a^n b^n c^n \mid n \geq 1\}$ je typu 1.

Definovali jsme *lineárně omezené automaty* (linear bounded automata, LBA) jako nedeterministické (!) Turingovy stroje, které mohou přepisovat pouze políčka pásky, na nichž je zapsáno vstupní slovo.

> Ekvivalentní definice je, že vstupní slovo je vždy napsáno mezi dvěma speciálními zarážkami, např. $\triangleright abaab \triangleleft$, které stroj nemůže přepsat a z $\triangleright$ nemůže doleva a z $\triangleleft$ nemůže doprava.

Dokázali jsme následující tvrzení.

Tvrzení 2 *Třída jazyků rozpoznávaných lineárně omezenými automaty je totožná s třídou jazyků typu* 1 *v Chomského hierarchii.*

Připomněli jsme, že jazyky typu 2, tzv. *bezkontextové jazyky*, jsou generovány *bezkontextovými gramatikami* (v nich je v každém pravidle $\alpha \to \beta$ levá strana pouhým jedním neterminálem, pravidla jsou tedy typu $X \to \beta$, kde $X \in \Pi$).

> Dá se ukázat, že nezkracující gramatiky generují tytéž jazyky jako tzv. *kontextové gramatiky*, v nichž jsou povolena pouze pravidla ve tvaru $\gamma X \delta \to \gamma\beta\delta$, kde $\beta \neq \varepsilon$. Proto ten název *kontextové jazyky* u jazyků typu 1.

Rovněž jsme připomněli, že bezkontextové gramatiky jsou ekvivalentní zásobníkovým automatům (co se týče třídy generovaných/rozpoznávaných jazyků). Obecně se pojmem zásobníkový automat myslí *nedeterministický* zásobníkový automat. Ukázali jsme si bezkontextový

jazyk, který není rozpoznáván žádným deterministickým zásobníkovým automatem (a není tedy tzv. deterministickým bezkontextovým jazykem). Podobná otázka u LBA, zda totiž existuje kontextový jazyk, který není rozpoznáván žádným deterministickým lineárně omezeným automatem, je dodnes otevřená.

Jazyky typu 3 jsou tzv. *regulární jazyky* (generované regulárními gramatikami, tedy gramatikami s pravidly typu $X \to wY$ a $X \to w$ kde X, Y jsou neterminály a w je řetězec terminálů). Víme, že tyto jazyky lze také zadat pomocí konečných automatů (deterministických či nedeterministických) nebo regulárních výrazů.

Diskutovali jsme uzavřenost tříd jazyků na množinové či jazykové operace. Speciálně pro doplněk jsme si uvědomili, že REG (třída regulárních jazyků, typ 3) je na něj uzavřena, ale CFL (třída bezkontextových jazyků, typ 2) není. Třída CSL (třída kontextových jazyků, typ 1) je na doplněk překvapivě uzavřena (diskutovali jsme tzv. Immerman–Szelepcsényi teorém). Pak jsme si ukázali, že třída RecEnum (tak budeme označovat třídu rekurzivně spočetných jazyků [recursively enumerable languages], tedy jazyků typu 0), na doplněk uzavřena není. To plyne z následující věty (nazývané někdy Postova věta)

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

a z faktu, že L_{HP} je částečně rozhodnutelný, ale ne rozhodnutelný. Tento fakt jsme si již ukázali dříve, větu jsme dokázali nyní.

> 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.

Speciálně jazyk $\overline{L_{HP}}$ (de facto odpovídající problému nezastavení Turingova stroje) není tedy (ani) částečně rozhodnutelný.

Uvědomili jsme si, že třída Rec (tj. třída rekurzivních jazyků, tedy jazyků rozhodovaných Turingovými stroji) je mezi CSL a RecEnum: CSL⊆Rec⊆RecEnum. Platnost inkluze Rec⊆RecEnum nám byla hned jasná, navíc díky L_{HP} víme, že tato inkluze je vlastní. Platnost inkluze CSL⊆Rec jsme si dokázali:

Tvrzení 4 *Je-li jazyk L rozpoznáván (nedeterministickým) lineárně omezeným automatem, pak je také rozhodován (deterministickým) Turingovým strojem.*

Příště si ukážeme, že i tato inkluze je vlastní.

Shrneme-li třídy jazyků, které jsme diskutovali, a vztahy mezi nimi, dostáváme:

$$\text{REG} \subset \text{CFL} \subset \text{CSL} \subset \text{Rec} \subset \text{RecEnum}.$$

(Symbolem $\subset$ označujeme vlastní inkluzi.) Umíme mj. snadno ukázat, že třídy REG a Rec jsou uzavřeny na doplněk, také umíme dokázat, že RecEnum na doplněk uzavřena není (např. $L_{HP} \in \text{RecEnum}$, ale $\overline{L_{HP}} \notin \text{RecEnum}$). O CFL víme, že na doplněk uzavřena není (např. pro $L = \{a^n b^n c^n \mid n \geq 1\}$ máme $\overline{L} \in \text{CFL}$, ale $\overline{\overline{L}} = L$ v CFL není), zatímco CSL uzavřena je (Immerman–Szelepcsényi).

Týden 4

Prodiskutovali jsme ukázku možného typu první zápočtové písemky. (Příslušný soubor je k dispozici na web-stránce předmětu.) Písemka se bude psát

ve středu 25.10.2017 od 15.00.

Předtím ve výuce od 14 hodin mj. zopakujeme dosavadní látku, jejíž pochopení má zápočtová písemka částečně prověřit. Horní limit pro odevzdání písemky bude 17.30, byť by písemka měla být zvládnutelná daleko dříve. (Jde o to, ať se teď nikdo necítí pod časovým tlakem. U opravných písemek v posledním týdnu bude již časový limit menší.)

Max. zisk za písemku je 20 bodů; stejně to bude i u druhé zápočtové písemky. K získání zápočtu je nutné v součtu získat alespoň 25 bodů.

Dokázali jsme, že inkluze CSL⊆Rec je vlastní, tedy:

Tvrzení 5 *Existuje jazyk, který je rozhodnutelný, ale není kontextový (tedy není jazykem typu* 1 *v Chomského hierarchii).*

Důkaz. Ukázali jsme, že to platí např. pro jazyk $L = \{\langle G\rangle \mid \langle G\rangle \notin L(G)\}$, kde symbolem G odkazujeme ke gramatikám typu 1 (tedy nezkracujícím gramatikám) s terminální abecedou $\{0,1\}$ a kde $\langle G\rangle$ je kód gramatiky G v nějakém přirozeném kódování gramatik řetezci z $\{0,1\}^*$. □

Pak jsme se bavili 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 6 *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 *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 $L_{P_1} \leq_\mathrm{m} L_{P_2}$.

Pak jsme dokázali několik jednoduchých, ale důležitých, faktů:

Tvrzení 7 *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 6 bylo prokázání, že HP$\leq_\mathrm{m}$UHP.

Pak jsme dokonce ukázali:

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

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.

Důležitým obratem v důkazu byl fakt, že výpočty Turingových strojů lze přirozeně popisovat řetězci ve vhodné abecedě; výpočet (nebo též "výpočetní historie") je prostě příslušná posloupnost konfigurací oddělených speciálním znakem: např. $\#C_0\#C_1\#C_2\#\cdots C_k\#$. Podstatou důkazu byl fakt, že když M se zastaví na w, tak existuje (konečný) řetězec $\#C_0\#C_1\#C_2\#\cdots C_k\#$, 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).

Z faktu co-HP$\leq_{\mathrm{m}}$UHP plyne, že UHP není (ani) částečně rozhodnutelný. Z faktu HP$\leq_{\mathrm{m}}$UHP plyne co-HP$\leq_{\mathrm{m}}$co-UHP (uvědomme si, že obecně platí $L_1 \leq_{\mathrm{m}} L_2 \Leftrightarrow \overline{L_1} \leq_{\mathrm{m}} \overline{L_2}$), a tedy ani doplňkový problém problému UHP (zformulujte si jej) není částečně rozhodnutelný.

> 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.

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. Ukázali jsme, že *co-HP* $\leq_{\mathrm{m}}$ All_{CFL}.

Opět jsme přitom použili výpočetní historie $\#C_0\#C_1\#C_2\#C_3\#\cdots C_k\#$, tentokrát ale ve formě $\#C_0\#(C_1)^R\#C_2\#(C_3)^R\#\cdots\#$, tedy "každou druhou" konfiguraci píšeme obráceně. Důvod je ten, že pak nedeterministický zásobníkový automat snadno odhalí, když zadaný konečný řetězec nepopisuje výpočet stroje M na w, který dospěje do koncového stavu.

Když se tedy M nezastaví na w, tak příslušný zásobníkový automat Z přijme všechny řetězce (v každém je schopen odhalit "chybu"); když se M zastaví na w, tak existuje řetězec, který Z nepřijímá (chybu v popisu konečného výpočtu M na w nenajde, protože tam žádná není). □

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: $L(G_1) = L(G_2)$?
(Co z toho plyne pro problém EQ_{CFL}?)

Týden 5

Připomněli jsme si problémy schématicky označené X_Y, kde $X \in \{Acc, Emp, All, EQ\}$ (postupně problém přijímání [acceptance], prázdnosti [emptiness], univerzality [ve smyslu přijímání všech slov] a ekvivalence) a $Y \in \{REG, CFL, CSL, Rec, RecEnum\}$, a jejich rozhodnutelnost či nerozhodnutelnost.

(Např. problémy EQ_{REG} a Emp_{CFL} jsou rozhodnutelné a např. problémy All_{CFL} a Emp_{CSL} rozhodnutelné nejsou.)

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), 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 věty. Uvažujme libovolnou netriviální vstupně/výstupní vlastnost V Turingových strojů. Nechť stroj M_1, který se nezastaví na žádný vstup, vlastnost V nemá, a nechť jistý stroj M_2 vlastnost V má, nebo naopak; takové dva stroje M_1, M_2 nutně musí existovat.

Uvažme nyní instanci M, w problému zastavení. Sestrojme k ní stroj M', který se chová takto: vstup u nejprve ignoruje a simuluje stroj M na w (slovo w má M' uloženo ve svém počátečním stavu, napíše ho na pásku napravo od vstupu u a na něm pak simuluje M; víme, že stačí uvažovat M, který nechodí nalevo od svého vstupu, takže kolize se vstupem u při této simulaci nenastane); pokud se tato simulace zastaví (tedy M se zastaví na w), tak M' smaže vše zbylé po této simulaci a pokračuje simulací stroje M_2 na vstupu u.

Vidíme, že platí: pokud M se zastaví na w, tak M' má stejné I/O chování (má stejnou I/O tabulku) jako M_2; pokud se M nezastaví na w, tak M' má stejné I/O chování (má stejnou I/O tabulku) jako M_1. Kdyby tedy vlastnost V byla rozhodnutelná, byl by rozhodnutelný i problém zastavení.

Ukázali jsme vlastně, že $\mathrm{HP} \leq_\mathrm{m} P_V$ nebo $\mathrm{HP} \leq_\mathrm{m} \text{co-}P_V$ (kde co-P_V je doplňkový problém problému P_V); o který z těchto případů se jedná, je určeno tím, zda stroj M_1 (který se nezastaví na žádný vstup) vlastnost V nemá či má. □

Enumerátory

Uvedli jsme pojem Turingova stroje jako *enumerátoru*: takový stroj má navíc "write-only" výstupní pásku a pro prázdný vstup má nekonečný výpočet, při němž čas od času přidá (neboli "vytiskne") nějaké slovo na výstupní pásku (ta by tedy po proběhnutí celého nekonečného výpočtu obsahovala případně nekonečnou posloupnost slov, řekněme vzájemně oddělených nějakým speciálním oddělovacím znakem).

Enumerátor $\mathcal{E}$ takto definuje jazyk $L(\mathcal{E})$, což je množina všech slov, která jsou "vytisknuta" na výstupní pásku v průběhu onoho nekonečného výpočtu.

Ukázali jsme si, že množina $\{L \mid L = L(\mathcal{E})$ pro nějaký enumerátor $\mathcal{E}\}$ je totožná s již dříve zavedenou množinou RecEnum. (Je teď jasnější i název "Recursively Enumerable", česky "rekurzivně spočetný".)

Mj. jsme si uvědomili, že množina všech tautologií výrokové logiky je rozhodnutelná (což znamená, že problém příslušnosti k množině tautologií VL je rozhodnutelný) a že množina všech tautologií (tedy logicky platných formulí) predikátové logiky je rekurzivně spočetná (neboli částečně rozhodnutelná) – díky větě o úplnosti predikátového kalkulu (rozhodnutelná ovšem tato množina není).

Věta o rekurzi

Diskutovali jsme i problém s *OwnCode* avízovaném v prvním týdnu, neboli *větu o rekurzi*. (Ta nebude předmětem zápočtové písemky.)

Týden 6

Větu o rekurzi jsme formálně zapsali takto:

Věta 10 *Nechť Turingův stroj T vyčísluje funkci $t : \Sigma^* \times \Sigma^* \to \Sigma^*$. Pak existuje Turingův stroj R, který vyčísluje funkci $r : \Sigma^* \to \Sigma^*$ splňující $\forall u \in \Sigma^* : r(u) = t(u, \langle R \rangle)$.*

Vyvodili jsme pomocí ní následující tvrzení o minimálních Turingových strojích; řekneme, že stroj M je minimální, jestliže neexistuje M', který vyčísluje tutéž funkci jako M a přitom platí $\langle M' \rangle < \langle M \rangle$.

Tvrzení 11 *Jazyk $L = \{\langle M \rangle \mid M$ je minimální $\}$ není rekurzivně spočetný.*

Důkaz. Kdyby jazyk L byl rekurzivně spočetný, pak by jej enumeroval nějaký enumerátor $\mathcal{E}$. Pak by ovšem následující TS M_1 (v jehož definici využíváme větu o rekurzi) vedl ke sporu:

M_1 pro vstup u nejdříve simuluje $\mathcal{E}$ a až ten (nutně) vygeneruje stroj M', který splňuje $\langle M' \rangle > \langle M_1 \rangle$, simuluje M' na u.

(M_1 tedy vyčísluje tutéž funkci jako M', což je spor s tím, že M' má být minimální.) □

Zopakovali jsme stručně pojmy potřebné k zápočtové písemce, která se pak psala.

Týden 7

Výsledky první zápočtové písemky, diskuse řešení.

Připomenutí věty o rekurzi a uvedení důsledku:

Tvrzení 12 *Nechť $t : \{0,1\}^* \to \{0,1\}^*$ je (totální) vyčíslitelná funkce. (Existuje tedy Turingův stroj, který ke každému $w \in \{0,1\}^*$ vypočítá $t(w)$.) Pak existuje Turingův stroj F, s kódem $\langle F\rangle \in \{0,1\}^*$, takový že $t(\langle F\rangle)$ popisuje Turingův stroj, který je ekvivalentní s F (tedy F má stejné I/O chování jako stroj popsaný slovem $t(\langle F\rangle)$). Zde slovo z $\{0,1\}^*$, které není standardním kódem Turingova stroje, chápeme např. jako popis Turingova stroje, který každý vstup ihned zamítne.*

Důkaz. Díky větě o rekurzi můžeme legálně definovat F takto:

Pro vstup u:

nejdříve získej $\langle F\rangle$, pak spočítej $t(\langle F\rangle)$ a simuluj stroj popsaný slovem $t(\langle F\rangle)$ na u. □

V Tvrzení 12 je vhodné přemýšlet o t jako o "transformaci programů" (k zadanému počítačovému programu vyrobí transformace jiný program, např. takový, který zadaný program vytiskne). O F můžeme přemýšlet jako o pevném bodu (fixed point) této transformace. (Není to ovšem pevný bod, který musel splňovat podmínku $t(\langle F\rangle) = \langle F\rangle$, jde jen o to, že I/O chování příslušných strojů je stejné.)

Poznámky k aplikacím v logice.

Alespoň letmo jsme diskutovali následujících fakta.

Věta 13 *Presburgerova aritmetika, tj. množina $Th(\mathbb{N}, Plus)$, je rozhodnutelná.*

Množinu $Th(\mathbb{N}, Plus)$ chápeme jako množinu těch uzavřených formulí predikátové logiky prvního řádu s ternárním predikátovým symbolem $Plus$, které jsou pravdivé ve standardním modelu s univerzem $\mathbb{N} = \{0, 1, 2, \dots\}$ při interpretaci $Plus(x, y, z)$ jako vztahu $x + y = z$.

Důkaz se dá elegantně provést využitím teorie konečných automatů (neprovedli jsme).

Následuje důležitý výsledek, který v 30. letech 20. století vzbudil velký rozruch:

Věta 14 *Množina $Th(\mathbb{N}, Plus, Mult)$ není rozhodnutelná.*

Množina $Th(\mathbb{N}, Plus, Mult)$ je definovaná analogicky jako $Th(\mathbb{N}, Plus)$, přičemž predikát $Mult(x, y, z)$ je interpretován jako vztah $x \cdot y = z$.

Podstatou důkazu nerozhodnutelnosti je, že ke každému Turingovu stroji M a jeho vstupu w lze algoritmicky sestrojit formuli $\Phi_{M,w}(x)$ v jazyce s $Plus$ a $Mult$ (v níž je x jediná volná proměnná) tak, že M se zastaví na w právě tehdy, když (uzavřená) formule $(\exists x)\Phi_{M,w}$ je pravdivá (ve standardním modelu $\mathbb{N}$, tj. je prvkem $Th(\mathbb{N}, Plus, Mult)$). Formule $\Phi_{M,w}(x)$ je konstruována tak, že de facto říká: číslo x je kódem výpočtu stroje M na w, který skončí v koncové konfiguraci. (Číslo x tedy kóduje příslušnou posloupnost konfigurací stroje M; podmínka, že posloupnost odpovídá instrukcím stroje M se dá vyjádřit pomocí sčítání a násobení, což vyžaduje kus technické víceméně rutinní práce.)

Další velmi důležitý výsledek se týká neúplnosti axiomatizací aritmetiky (první Gödelova věta o neúplnosti):

Věta 15 *Každá axiomatizace struktury* $(\mathbb{N}, Plus, Mult)$ *v jazyce predikátové logiky prvního řádu, která je korektní (tj. všechny její uzavřené dokazatalné formule patří do* $Th(\mathbb{N}, Plus, Mult)$*) a rozhodnutelná (tj. existuje algoritmus, který rozhoduje, zda daná formule je axiomem), je neúplná, což znamená, že existuje formule v* $Th(\mathbb{N}, Plus, Mult)$*, která není dokazatelná (neboli existuje pravdivá formule, která není dokazatelná).*

V důkazu lze využít výše zmíněné formule $\Phi_{M,w}(x)$ a větu o rekurzi, a dále očividný fakt, že pro každou uvedenou axiomatizaci existuje enumerátor $\mathcal{E}$, který generuje všechny uzavřené dokazatelné formule. (V takové axiomatizaci je množina důkazů rozhodnutelná a množina dokazatelných formulí částečně rozhodnutelná.)

Konkrétně můžeme sestrojit Turingův stroj S, který se pro každý vstup chová následovně:

Získej svůj kód $\langle S \rangle$ a sestav formuli $\neg(\exists x)\Phi_{S,0}$ (která říká "Stroj S se nezastaví na vstup 0"). Spusť enumerátor $\mathcal{E}$; pokud ten někdy vygeneruje onu formuli $\neg(\exists x)\Phi_{S,0}$, zastav se.

Je zřejmé, že program S se na vstup 0 nemůže zastavit (jinak bychom dostali spor s korektností axiomatizace). Formule $\neg(\exists x)\Phi_{S,0}$ je tedy pravdivá (patří do $Th(\mathbb{N}, Plus, Mult)$), ale není v dané axiomatizaci dokazatelná. (Když ji přidáme jako další axiom, enumerátor $\mathcal{E}$ a tím i stroj S se příslušně změní a dostaneme opět pravdivou nedokazatelnou formuli $\neg(\exists x)\Phi_{S',0}$, kde S' je onen "změněný S".)

Teorie (výpočetní) složitosti

K Turingově stroji M jsme zavedli jeho časovou složitost jako funkci $T_M : \mathbb{N} \to \mathbb{N}$ .....

Připomněli jsme mj. značení O, o, Θ, které porovnává funkce z hlediska jejich asymptotického růstu.

Týden 8

Dokázali jsme:

Tvrzení 16 *Vícepáskový Turingův stroj M s časovou složitostí T_M lze simulovat jednopáskovým Turingovým strojem M' s časovou složitostí $T_{M'}(n) \in O((T_M(n))^2)$.*

Zavedli jsme třídy problémů $\textsc{Time}\,(f(n))$ a definovali

$$\text{PTIME} = \bigcup_{k \in \mathbb{N}} \textsc{Time}\,(n^k).$$

Všimli jsme si *robustnosti* třídy PTIME (díky Tvrzení 23 a vlastnostem polynomů).

Připomněli jsme si polynomiální algoritmy pro problémy

- dosažitelnost v grafu,
- nejkratší cesta v grafu,
- minimální kostra grafu,
- třídění,
- ...

Tím jsme připomněli i algoritmické techniky *prohledávání*, *rozděl a panuj*, *dynamické programování*. Metodu dynamického programování jsme si ilustrovali i předvedením polynomiálního algoritmu pro

- problém příslušnosti slova k danému bezkontextovému jazyku (reprezentovanému gramatikou v Chomského normální formě).

Problémy SAT, 3-SAT, IS_{dec} (Independent Set), VC_{dec} (Vertex Cover), HC (hamiltonovský cyklus), HK (hamiltonovská kružnice), TSP_{dec} (Travelling Salesman Problem) jsme si definovali a ilustrovali. O těch se neví, zda je lze řešit polynomiálními algoritmy.

Zavedli jsme relaci $\leq_\text{p}$ (polynomiální redukovatelnost, či polynomiální převeditelnost) na množině rozhodovacích problémů (nebo jím odpovídajících jazyků). Ze "šňůry"

$$\text{SAT} \leq_\text{p} \text{3-SAT} \leq_\text{p} \text{IS}_{dec} \leq_\text{p} \text{VC}_{dec} \leq_\text{p} \text{HC} \leq_\text{p} \text{HK} \leq_\text{p} \text{TSP}_{dec}$$

jsme zatím ukázali $\text{IS}_{dec} \leq_\text{p} \text{VC}_{dec}$, $\text{HC} \leq_\text{p} \text{HK}$ a $\text{HK} \leq_\text{p} \text{TSP}_{dec}$ (a navíc $\text{HK} \leq_\text{p} \text{HC}$ a $\text{VC}_{dec} \leq_\text{p} \text{IS}_{dec}$).

Také jsme definovali Clique_{dec} (problém kliky v grafu) a uvědomili si $\text{IS}_{dec} \leq_\text{p} \text{Clique}_{dec}$ a $\text{Clique}_{dec} \leq_\text{p} \text{IS}_{dec}$. Také jsme snadno odvodili:

Tvrzení 17 *Relace $\leq_\text{p}$ je tranzitivní.*

Tvrzení 18 *Jestliže $\mathcal{P}_1 \leq_\text{p} \mathcal{P}_2$ a $\mathcal{P}_2 \in \text{PTIME}$, pak $\mathcal{P}_1 \in \text{PTIME}$.*

Týden 9

Dokončili jsme demonstraci polynomiálních redukcí ve "šňůře"

$$\text{SAT} \leq_\text{p} \text{3-SAT} \leq_\text{p} \text{IS}_{dec} \leq_\text{p} \text{VC}_{dec} \leq_\text{p} \text{HC} \leq_\text{p} \text{HK} \leq_\text{p} \Delta\text{-TSP}_{dec} \leq_\text{p} \text{TSP}_{dec}, \qquad (1)$$

do níž jsme přidali podproblém Δ-TSP$_{dec}$ problému TSP$_{dec}$, kde "trojúhelník" Δ označuje, že uvažujeme jen instance splňující trojúhelníkovou nerovnost.

Definovali jsme, co je složitost $T_M : \mathbb{N} \to \mathbb{N}$ pro *nedeterministický* Turingův stroj M (jenž rozhoduje nějaký problém tím, že pro pozitivní instanci má alespoň jeden výpočet akceptující a pro negativní instanci má všechny výpočty zamítající).

Zavedli jsme třídy problémů $\text{NTIME}\,(f(n))$ a definovali

$$\text{NPTIME} = \bigcup_{k \in \mathbb{N}} \text{NTIME}\,(n^k).$$

Třída PTIME se také zkráceně označuje P a místo NPTIME také píšeme NP. Triviálně platí:

Tvrzení 19 $\text{P} \subseteq \text{NP}$.

Otázka, zda P = NP, je snad nejdůležitější problém teorie složitosti. Obecně panuje názor, že P $\neq$ NP, ale nikdo to neumí dokázat.

My jsme si připomněli, že $\leq_\text{p}$ je tranzitivní, a uvědomili si, že také platí

Tvrzení 20 *Jestliže* $\mathcal{P}_1 \leq_\text{p} \mathcal{P}_2$ *a* $\mathcal{P}_2 \in \text{NP}$, *pak* $\mathcal{P}_1 \in \text{NP}$.

Ukázali jsme, že TSP$_{dec}$ patří do NP (a tedy mj. všechny problémy ve šnůře (1) jsou v NP). Načrtli jsme prostě pseudokód jednoduchého nedeterministického algoritmu, který TSP$_{dec}$ rozhoduje a má očividně polynomiální složitost.

Pak jsme uvedli důležitou definici:

- (Rozhodovací) problém $\mathcal{P}$ je NP-*těžký*, jestliže $\forall \mathcal{P}' \in \text{NP} : \mathcal{P}' \leq_\text{p} \mathcal{P}$.
- (Rozhodovací) problém je NP-*úplný*, jestliže je NP-těžký a patří do NP.

Mj. jsme si všimli:

Tvrzení 21 *Jestliže* $\mathcal{P}_1 \leq_\text{p} \mathcal{P}_2$ *a* $\mathcal{P}_1$ *je* NP-*těžký, pak* $\mathcal{P}_2$ *je* NP-*těžký.*

Příště také dokážeme následující důležitou větu:

Věta 22 *(Cook) Problém SAT je NP-úplný.*

Vyvodíme tedy mj., že všechny problémy ve šnůře (1) jsou NP-úplné.

Na cvičení jsme procvičovali formulaci problémů (ze "šnůry"), speciálně rozhodovacích verzí optimalizačních problémů, příklady pozitivních a negativních instancí u jednotlivých problémů, prokazování příslušnosti problémů k NP (návrhem nedeterministických polynomiálních algoritmů), ...

Týden 10

Na cvičení jsme prodiskutovali ukázku možného typu druhé zápočtové písemky. (Příslušný soubor je k dispozici na web-stránce předmětu.) Písemka se bude psát

ve středu 6.12.2017 od 15.00.

Předtím ve výuce od 14 hodin mj. zopakujeme dosavadní látku, jejíž pochopení má zápočtová písemka částečně prověřit. Horní limit pro odevzdání písemky bude 17.30, byť by písemka měla být zvládnutelná daleko dříve. (Jde o to, ať se teď nikdo necítí pod časovým tlakem. U opravných písemek v posledním týdnu bude již časový limit menší.)

Max. zisk za písemku je 20 bodů, stejně jako u první zápočtové písemky. K získání zápočtu je nutné v součtu získat alespoň 25 bodů.

==========

Dokázali jsme Cookovu větu (SAT je NP-úplný). Neprovedli jsme důkaz formálně, ale načrtli si myšlenku popisu výpočtu nedeterministického Turingova stroje (kde výpočet je chápán jako posloupnost konfigurací zapsaných "pod sebou", tedy každá konfigurace je de facto jeden řádek dvourozměrné tabulky) pomocí booleovské formule (která je konjunkcí malých "lokálních" podmínek). Takto je vidět, že každý problém $\mathcal{P}$ z NP (tedy rozhodovaný nedeterministickým Turingovým strojem s polynomiální časovou složitostí) lze polynomiálně převést na SAT (tedy platí $\mathcal{P} \leq_{\mathrm{p}} \mathrm{SAT}$).

Silná a slabá NP-obtížnost v případě instancí s čísly.

Často jsou součástí instancí problémů čísla (např u TSP). Některé problémy jsou NP-těžké i při omezení se na malá čísla v instancích či, přesněji řečeno, i při zápisu čísel unárně (počtem "čárek"); např. se jedná o TSP_{dec} (jak jsme si vlastně ukázali při redukci $\mathrm{HK} \leq_{\mathrm{p}} \mathrm{TSP}_{dec}$). Takové problémy se také nazývají *silně* NP-*těžké*.

Existují ale i problémy, u nichž jsou velikost čísel a jejich "stručný" (tedy binární či dekadický) zápis pro NP-obtížnost podstatné. Např se jedná o problém SUBSET-SUM (speciální případ problému batohu): instancí je množina čísel $\{d_1, d_2, \ldots, d_k\}$ a číslo ℓ; otázkou je, zda existuje "výběr" $\{i_1, i_2, \ldots, i_m\} \subseteq \{1, 2, \ldots, k\}$ tak, že $\sum_{j=1}^{m} d_{i_j} = \ell$. V případě unárního zápisu čísel d_i a ℓ existuje polynomiální rozhodovací algoritmus (připomněli jsme si, že je založen na dynamickém programování); také říkáme, že SUBSET-SUM je řešitelný *pseudo-polynomiálním* algoritmem (kde předpona "pseudo" odkazuje k onomu unárnímu kódování čísel).

Při standardním (binárním či dekadickém) zápisu čísel není pro SUBSET-SUM polynomiální algoritmus znám. Ukázali jsme si konstrukci prokazující

$$\text{3-SAT} \leq_{\mathrm{p}} \text{SUBSET-SUM}$$

a vyvodili tak, že SUBSET-SUM je NP-úplný (jeho příslušnost k NP je zřejmá). Je to tedy příklad *slabě* NP-*těžkého problému* (je sice NP-těžký, ale při unárním zápisu čísel je polynomiální).

Pohovořili jsme o otázce $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$.

Uvedli jsme i třídu co-NP a diskutovali $\mathrm{NP} \cap \text{co-NP}$.

Také jsme zmínili nejasný status problému grafového izomorfismu.

Paměťová (či prostorová) složitost

Zavedli jsme paměťovou (či prostorovou) složitost $S_M : \mathbb{N} \to \mathbb{N}$ Turingova stroje M a definovali třídy problémů $\textsc{Space}\,(f(n))$, $\textsc{NSpace}\,(f(n))$, a dále třídy PSPACE a NPSPACE.

Avizovali jsme, že je známo

$$\text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE} = \text{NPSPACE}.$$

Dosud není vyloučena dokonce ani rovnost PTIME=PSPACE, byť vypadá "velmi nepravděpodobně". Příště se k třídám prostorové složitosti vrátíme podrobněji.

Týden 11

Začali jsme "oblázkovou hrou":

> Problém MinPebble$_{dec}$
>
> Instance: Acyklický orientovaný graf G, vrchol v a číslo k, což je počet oblázků.
>
> Otázka: Dá se docílit situace s oblázkem na vrcholu v v následující "hře pro jednoho hráče"? Podle pravidel hry lze opakovaně provádět následující "tahy":
>
> - lze položit oblázek na vrchol u, jestliže na všech vrcholech, z nichž vede hrana do u, leží oblázky;
> - pokud na (kterémkoli) vrcholu leží oblázek, lze jej odebrat.

Uvědomili jsme si souvislost např. s dostačujícím počtem registrů při vyhodnocování aritmetického výrazu.

Podařilo se nám najít nedeterministický algoritmus, který daný problém rozhoduje (v kladném případě alespoň jeden výpočet akceptuje a v záporném případě všechny výpočty zamítají) a pracuje pouze v polynomiálním (dokonce jen lineárním) prostoru.

> Abychom zajistili, že každý výpočet je konečný, omezili jsme počet tahů v oblázkové hře na $2^{|V|}$, kde V je množina vrcholů. Při nejkratším běhu hry, který skončí položením oblázku na v (existuje-li takový běh), se totiž neopakuje konfigurace, tj. rozmístění oblázků na grafu. Hodnotu čítače tahů lze pochopitelně uchovávat v lineárním prostoru; při binárním zápisu postačí $\log 2^{|V|}$ bitů, tedy $|V|$ bitů.

Prokázali jsme tak, že MinPebble$_{dec}$ patří do NPSPACE. Při úvaze, zda se tento horní odhad dá zlepšit, jsme si nejdříve připomněli, že "nedeterministický čas" umíme simulovat deterministicky jen s exponenciální ztrátou; přesněji řečeno:

Tvrzení 23 *Nedeterministický Turingův stroj M s časovou složitostí T_M (kde $T_M(n) \geq n$) lze simulovat deterministickým Turingovým strojem M' s časovou složitostí $T_{M'}(n) \in 2^{O(T_M(n))}$.*

Zlepšit to neumíme, tak se zdá, že platí PTIME $\neq$ NPTIME (dokázat to ovšem neumíme). Pro nedeterministický prostor jsme ale dokázali Savitchovu větu:

Věta 24 *(Savitch) Pro každou funkci $f : \mathbb{N} \to \mathbb{N}$ splňující $f(n) \geq n$ platí*

$$\text{NSpace}\,(f(n)) \subseteq \text{Space}\,((f(n))^2).$$

Z toho mj. plyne PSPACE = NPSPACE.

Hlavním obratem v důkazu bylo, že pro nedeterministický Turingův stroj M pracující na slově w v prostoru $f(|w|)$ lze existenci/neexistenci přijímajícího výpočtu zjistit provedením $CanYield(C_0, C_{acc}, 2^{d \cdot f(|w|)})$, kde C_0 je počáteční konfigurace stroje M na w, C_{acc} je přijímající konfigurace

> (pro jednoduchost uvažujeme jen M, který "po sobě uklidí", tj. před koncem výpočtu smaže symboly na pásce a hlavu posune na nejlevější políčko původního vstupu; takto je přijímající konfigurace určena jednoznačně),

d je konstanta zaručující, že počet konfigurací stroje M velikosti nejvýše $f(|w|)$ je nejvýše $2^{d \cdot f(|w|)}$, a (rekurzivní) booleovská procedura $CanYield$ je definována takto:

> procedure $CanYield(C_1, C_2, t)$:
>
> (kde C_1, C_2 jsou konfigurace stroje M velikosti $f(|w|)$ a t je kladné celé číslo)
>
> pro $t = 1$ vrať TRUE, jestliže M umožňuje přejít jedním krokem z C_1 do C_2, a jinak vrať FALSE
>
> pro $t > 1$ projdi všechny konfigurace C velikosti $f(|w|)$ a pokud pro některou z nich platí
>
> $$CanYield(C_1, C, \tfrac{t}{2}) \text{ a zároveň } CanYield(C, C_2, \tfrac{t}{2})$$
> (kde při dělení t zaokrouhlujeme nahoru),
>
> vrať TRUE.
>
> Jinak (pokud pro žádnou konfiguraci C ona konjunkce neplatí) vrať FALSE.

Snadnou analýzou jsme zjistili, že pro provedení $CanYield(C_0, C_{acc}, 2^{d \cdot f(|w|)})$ stačí prostor $O((f(|w|))^2)$.

> Rychle jsme diskutovali i fakt, že uvedený důkaz Savitchovy věty tiše předpokládá, že funkce f je tzv. "prostorově konstruovatelná" (což všechny "běžné" funkce jsou). Pro zcela obecnou funkci se důkaz ještě musí doplnit.

Jelikož jsme dokázali, že MinPebble$_{dec}$ patří do NPSPACE, víme teď, že patří i do PSPACE (neboť PSPACE = NPSPACE).

Analogicky jako u NP-úplnosti jsme zavedli PSPACE-úplnost.

> Uvědomili jsme si, proč i zde používáme redukci $\leq_{\mathrm{p}}$, kde index p odkazuje na polynomiální *čas*.

Odvodili jsme mj.:

Tvrzení 25 *Jestliže $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ a $\mathcal{P}_1$ je* PSPACE-*těžký, pak $\mathcal{P}_2$ je* PSPACE-*těžký.*

Dokázali jsme analogii Cookovy věty pro PSPACE:

Věta 26 *Problém QBF je* PSPACE-*úplný.*

Problém QBF se ptá, zda daná *plně kvantifikovaná* booleovská formule je pravdivá.

Příslušnost k PSPACE jsme prokázali přímočarým rekurzivním algoritmem (který předpokládá zadanou formuli v prenexní formě).

PSPACE-obtížnost jsme dokázali úpravou obratu, který jsme použili v důkazu Savitchovy věty:

Pro daný (deterministický) Turingův stroj M pracující na w v prostoru omezeném $pol(|w|)$ (kde pol je určitý polynom)

zkonstruujeme formuli označenou $\phi_{C_1,C_2,0}$ (bez kvantifikátorů), která má sadu booleovských proměnných popisujicích dvě konfigurace velikosti $pol(|w|)$, označené C_1, C_2, a která je splněna pravdivostním ohodnocením proměnných právě tehdy, když toto ohodnocení popisuje dvě konfigurace, z nichž druhá je přímým následníkem první při výpočtu stroje M.

Dále zkonstruujeme $d \cdot pol(|w|)$ dalších formulí označených $\phi_{C_1,C_2,1}$, $\phi_{C_1,C_2,2}$, $\phi_{C_1,C_2,3}$, $\phi_{C_1,C_2,4}$, $\cdots$, $\phi_{C_1,C_2,d\cdot pol(|w|)}$

> (kde d je konstanta zaručující, že počet konfigurací velikosti nejvýš $pol(|w|)$ je nejvýš $2^{d\cdot pol(|w|)}$)

tak, že $\phi_{C_1,C_2,t}$ je ve tvaru

$$(\exists C)(\forall C', C'') : \big(((C' \leftrightarrow C_1) \wedge (C'' \leftrightarrow C)) \vee ((C' \leftrightarrow C) \wedge (C'' \leftrightarrow C_2))\big) \rightarrow \phi_{C',C'',t-1}.$$

Je vidět, že M přijímá w právě tehdy, když formule $\phi_{C_0,C_{acc},d\cdot pol(|w|)}$ je pravdivá. Pro fixní M má tato formule velikost $O((pol(|w|))^2)$.

> Výrazem $\phi_{C_0,C_{acc},d\cdot pol(|w|)}$ pochopitelně myslíme formuli, která vznikne z $\phi_{C_1,C_2,d\cdot pol(|w|)}$ dosazením pravdivostních hodnot za volné proměnné tak, že první popsaná konfigurace je počáteční konfigurace stroje M na w a druhá je akceptující "uklizená" konfigurace.

Uvedli jsme překvapivý výsledek, že naše "solitérní" oblázková hra je PSPACE-úplná; platí totiž QBF$\leq_{\mathrm{p}}$ MinPebble$_{dec}$, byť jsme si technický důkaz tohoto faktu neukázali.

Typické PSPACE-úplné (či PSPACE-těžké) jsou problémy formulovatelné jako hry dvou hráčů, jako je například "generalizovaná geografie" GG. Důkaz QBF$\leq_{\mathrm{p}}$GG jsme dotáhli na cvičení.

Na cvičení jsme také ukázali, že problém NFA$_{\mathrm{EQ}}$ (ekvivalence nedeterministických konečných automatů) je v NPSPACE, a tedy i v PSPACE, neboť NPSPACE = PSPACE. Problém NFA$_{\mathrm{EQ}}$ je také PSPACE-úplný, byť jeho PSPACE-obtížnost jsme nedokazovali.

Týden 12

K dosud známým třídám složitosti jsme přidali třídy L (neboli LOGSPACE), NL (neboli NLOGSPACE), EXPTIME, NEXPTIME, EXPSPACE.

Rovnost EXPSPACE = NEXPSPACE jsme si hned uvědomili díky Savitchově větě. Rovnost L = NL z ní ovšem neplyne (a ne/platnost této rovnosti není známa). Turingovým strojem pracujícím v logaritmickém prostoru totiž chápeme stroj, který má read-only vstupní pásku (hlava na ní "běhá", ale jen čte) a dále pracovní pásku, na níž může i přepisovat, ale jejíž délka je omezena $O(\log n)$ (kde n je velikost vstupu).

Při definici NL-úplného problému se musíme omezit na tzv. logspace-redukce:

$\mathcal{P}_1 \leq_{\mathrm{L}} \mathcal{P}_2$ znamená, že existuje algoritmus s logaritmickou prostorovou složitostí, který k instanci I_1 problému $\mathcal{P}_1$ vyrobí instanci I_2 problému $\mathcal{P}_2$ se stejnou odpovědí na příslušné otázky. Redukující Turingův stroj zde má read-only vstupní pásku (s napsanou I_1), write-only výstupní pásku (kam píše I_2) a dále pracovní pásku, na níž může i přepisovat, ale jejíž délka je omezena $O(\log n)$, kde n je velikost instance I_1.

Aspoň jsme si řekli, že problém dosažitelnosti v orientovaném grafu je NL-úplný (příslušnost k NL jsme ovšem snadno nahlédli). Víme, že tento problém je v PTIME, a tedy platí i NL $\subseteq$ PTIME.

Také jsme diskutovali, že šikovnou úpravou diagonalizační metody lze ukázat, že inkluze PTIME $\subseteq$ EXPTIME, PSPACE $\subseteq$ EXPSPACE, a také NL $\subseteq$ PSPACE, jsou vlastní.

> Víme tedy, že v řetězci L $\subseteq$ NL $\subseteq$ PTIME $\subseteq$ NPTIME $\subseteq$ PSPACE jsou některé inkluze vlastní, ale nevíme, které.

Připomněli jsme také, že student Szelepcsényi před třiceti lety dokázal mj. i překvapivý fakt NL = co-NL (což plyne z věty dnes nazývané Immerman–Szelepcsényi teorém).

> Pak se psala druhá zápočtová písemka.

Týden 13

14:00 - 14:50: opravené písemky, zopakování látky, diskuse průběhu zkoušky.

15:00 - 16:15: oprava první zápočtové písemky

16:15 - 17:30: oprava druhé zápočtové písemky

(Minimum k získání zápočtu bylo sníženo: v součtu stačí 21 bodů.)

Seznam otázek k ústní zkoušce.

A) Vyčíslitelnost

1. Pojem *algoritmus* a jeho reprezentace Turingovým strojem; Church-Turingova teze. Varianty Turingových strojů (oboustranně/jednostranně nekonečná páska, více pásek) a jejich vzájemné simulace.
2. Kódování Turingova stroje řetězcem v abecedě $\{0, 1\}$. Univerzální Turingův stroj (popis jeho činnosti).
3. Pojem *problém* (neboli výpočetní problém). Rozhodovací (tj. ANO/NE) problémy; jejich prezentace jako jazyků v nějaké abecedě. Rozhodování problémů Turingovými stroji a také jejich rozhodování *nedeterministickými* Turingovými stroji. Simulace nedeterministického Turingova stroje deterministickým.
4. Rozhodnutelné a částečně rozhodnutelné problémy. Souvislost s doplňkovými problémy. Turingovy stroje jako enumerátory.
5. Chomského hierarchie jazyků. Charakterizace jejích čtyř tříd pomocí gramatik a příslušných "automatů". (Ne)uzavřenost jednotlivých tříd vůči doplňku.
6. Zařazení třídy tzv. rekurzivních jazyků mezi třídy Chomského hierarchie (s důkazem).
7. Nerozhodnutelné problémy. Důkazy diagonalizační metodou.
8. Redukce, neboli algoritmická převeditelnost, mezi problémy, tedy relace $\leq_{\mathrm{m}}$ ("mapping reduction", nebo také "many-one reduction"). Využití pro dokazování rozhodnutelnosti a nerozhodnutelnosti problémů.
9. Důkaz nerozhodnutelnosti ekvivalence bezkontextových gramatik.
10. Riceova věta; její důkaz a aplikace.
11. (Rozšiřující.) Věta o rekurzi. Aplikace při důkazu nerozhodnutelnosti "aritmetiky" a problému minimality Turingových strojů.

B) Složitost

1. Časová složitost Turingova stroje. Simulace vícepáskového stroje jednopáskovým s (nanejvýš) polynomiální ztrátou vzhledem k časové složitosti.
2. Časová složitost nedeterministického Turingova stroje. Simulace nedeterministického Turingova stroje deterministickým s (nanejvýš) exponenciální ztrátou vzhledem k časové složitosti.
3. Prostorová (neboli paměťová) složitost (ne)deterministického Turingova stroje. Simulace nedeterministického Turingova stroje deterministickým s (nanejvýš) polynomiální ztrátou vzhledem k prostorové složitosti (Savitchova věta).
4. Třídy časové a prostorové složitosti; jejich vzájemný vztah. Speciálně třídy PTIME, NPTIME, PSPACE, NPSPACE; robustnost jejich definice. Důkaz příslušnosti problému SAT a rozhodovací verze zvoleného optimalizačního problému k NPTIME.
5. Příklady problémů s polynomiální (časovou) složitostí a myšlenky příslušných algoritmů (dosažitelnost v grafu, nejkratší cesta mezi vrcholy, minimální kostra, problém příslušnosti k danému bezkontextovému jazyku, ...).

6. NP-úplnost. Využití polynomiálních redukcí (relace $\leq_{\mathrm{p}}$) k důkazům příslušnosti k PTIME či NPTIME a k důkazům NP-obtížnosti konkrétních problémů. Příklady konkrétních polynomiálních redukcí.
7. Cookova věta (SAT je NP-úplný) a myšlenka jejího důkazu. Otázka $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$.
8. Silná a slabá NP-úplnost u problémů, v nichž se vyskytují čísla; příklady příslušných problémů.
9. PSPACE-úplnost. Příklady PSPACE-úplných problémů. Prokázaní, že problém QBF (pravdivost plně kvantifikovaných booleovských formulí) je v PSPACE.
10. Důkaz příslušnosti $\mathrm{NFA}_{\mathrm{EQ}}$ (ekvivalence nedeterministických konečných automatů) k NPSPACE.
11. (Rozšiřující.) Třídy L a NL. NL-úplné problémy (vzhledem k relaci $\leq_{\mathrm{L}}$). Třídy EXPTIME, NEXPTIME, EXPSPACE, NEXPSPACE. Třídy doplňkových problémů (např. co-NL, co-NP).

Poznámky k průběhu zkoušky:
bez splněného zápočtu nelze jít na zkoušku; přihlašování na termíny vypsané v IS STAG;
zkoušení bude probíhat ve zkouškovém období ZS; zkouška bude ústní;
student si vytáhne dvě otázky, jedna bude z okruhu A) 1–10, druhá z B) 1–10;
prověřuje se také schopnost demonstrace pojmů a tvrzení konkrétními příklady ...;
pojmy z (rozšiřujících) otázek A.11 a B.11 budou diskutovány s adepty na hodnocení "A";
čas na písemnou přípravu: minimálně 15 minut;
čas na ústní zkoušení: zhruba 20 minut či více (podle situace).