# Vyčíslitelnost a složitost

(bc. předmět KMI/VYSLO)

Petr Jančar

5. prosince 2023

**Webová stránka předmětu**

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

**Poznámky k textu.** Tento text vzniká v průběhu kurzu v zimním semestru 2023/24 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. 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 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).

Avizoval jsem první zápočtovou písemku v 6. týdnu semestru 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 $\Sigma$ (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 $\Sigma$. 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, $\Sigma$ je konečná vstupní abeceda, $\Gamma$ 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 $\delta$, 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 $\delta$). Výpočet může skončit v koncové konfiguraci, tj. v konfiguraci $uqv$, kde $q \in F$, nebo být nekonečný.

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.* Diskutovali jsme konstrukce jednoduchých Turingových strojů, např. Turingova stroje rozhodujícího (nebezkontextový) jazyk $\{wcw \mid w \in \{a, b\}^*\}$.

## 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 $\delta$). 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 $\textsc{cod}$ 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.

Podrobněji jsme probrali 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) jsme navrhli 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 $\Gamma$ přidáme "náhradní (zapisovací) prázdný symbol" $\Box'$ a funkci $\delta$ příslušně upravíme.)

Při konstrukci $M'$ položíme $\Gamma' = \Gamma \cup \{¢\}$, kde $¢ \notin \Gamma$ je speciální symbol sloužící pro označení začátku pásky, a $Q'$ 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. $q \in Q$.

Funkce $\delta'$ pak vznikne rozšířením funkce $\delta$ takto:

(chceme, aby pro konfiguraci $q_0'w$, $w \in \Sigma^*$, platilo $q_0'w \vdash_{M'}^* ¢q_0w$)
$\delta'(q_0', \square) = (q_0, ¢, +1)$
$\delta'(q_0', x) = (q_x, ¢, +1)$ 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. $x \in \Sigma$
$\delta'(q_L, ¢) = (q_0, ¢, +1)$
(pro konfiguraci $q¢v$, $q \in Q$ [hlava přišla na ¢ ve stavu $q$], chceme docílit $q¢v \vdash_{M'}^* ¢q\square v$)
$\delta'(q, ¢) = (s_q, ¢, +1)$
$\delta'(s_q, \square) = (q, \square, 0)$
$\delta'(s_q, x) = (s_{(x,q)}, \square, +1)$ 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. $x \in \Gamma \setminus \square$
$\delta'(s_{(L,q)}, \square) = (q, \square, 0)$

Pak jsme se zamysleli, jak model dvouhlavých Turingových strojů simulujeme 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.* Dále jsme promýšleli simulace, mj. $k$-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 $\Sigma$ (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 $\Gamma = \{0, 1, \Box\}$ (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 $\{0,1,\Box\}$). (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.* Na cvičení jsme mj. dali dohromady 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

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

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 $q_0 w \vdash_M^* u q_- v$ 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 $\{0, 1, \Box\}$, 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 $\{0, 1, \Box\}$ se pak dá rutinně převést, jak již víme. $\Box$

**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.* Detailněji jsme diskutovali nové pojmy a tvrzení.

## Týden 5

**Nedeterministické Turingovy stroje.**
Před navázáním na minulou látku jsme udělali odbočku k nedeterministickým Turingovým strojům. Díky zkušenosti s nedeterministickými konečnými automaty a nedeterministickými zásobníkovými automaty jsme *nedeterministický Turingův stroj* navrhli jako

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \{q_+, q_-\}),$$

kde přechodová funkce je typu $\delta : (Q \setminus \{q_+, q_-\}) \times \Gamma \to 2^{Q \times \Gamma \times \{-1,0,+1\}}$ (např. tedy můžeme mít $\delta(q, a) = \{(q', a', +1), (q'', a'', -1)\}$). Definici relace $\vdash_M$ na množině konfigurací uvedenou u standardního (tedy *deterministického*) Turingova stroje přirozeně rozšíříme pro nedeterministický stroj:

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

Tedy obecně pro slovo $w \in \Sigma^*$ nemáme jen jediný výpočet, ale může jich být více (přičemž některé mohou být konečné a jiné nekonečné). Tyto výpočty lze přirozeně prezentovat stromem výpočtů, v němž každá větev odpovídá jednomu z možných výpočtů (iniciální konfigurace $q_0w$ označuje kořen stromu, listy jsou pak označeny příslušnými koncovými konfiguracemi).

I u nedeterministického Turingova stroje $M$ definujeme *jazyk*

$$L(M) = \{w \in \Sigma^* \mid q_0w \vdash_M^* uq_+v \text{ pro nějaké } u, v \in \Gamma^*\},$$

o němž říkáme, že je strojem $M$ *přijímán. Slovo* $w$ je tedy strojem $M$ *přijato*, jestliže pro něj existuje alespoň jeden akceptující výpočet. (Slovo není přijato, jestliže všechny výpočty pro něj jsou zamítající nebo nekonečné.)

Uvědomili jsme si, že strom výpočtů nedeterministického stroje $M_1$ na slově $w$ je možné konstruovat do šířky deterministickým strojem $M_2$. Z toho snadno plyne:

**Tvrzení 7** *Jazyk je přijímán nedeterministickým Turingovým strojem právě tehdy, když je přijímán deterministickým Turingovým strojem.*

Nedeterministické Turingovy stroje tedy přijímají právě částečně rozhodnutelné jazyky.

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

**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 8** $P \leq_{\mathrm{m}} P' \Leftrightarrow \text{co-}P \leq_{\mathrm{m}} \text{co-}P'$.

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 9** *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 $\Delta$; 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 10** *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 9 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 $M$ na $w$, lze provést nedeterministickým zásobníkovým automatem. Speciálně 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.)

To, že $All_{CFL}$ není ani částečně rozhodnutelný, teď snadno vyvodíme díky tomu, že doplňkový problém problému $All_{CFL}$ částečně rozhodnutelný je. (Proč?) □

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 $EQ_{CFL}$?)

**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 $L_V = \{\langle M \rangle \mid M$ 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).

*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?

## Týden 6

**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 HP$\leq_{\mathrm{m}} P_V$ nebo co-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$\leq_{\mathrm{m}} P_V$. □

Pak se psala **1. zápočtová písemka**.

## Týden 7

Výsledky první zápočtové písemky, diskuse řešení.

## Teorie (výpočetní) složitosti

**Časová složitost Turingova stroje, a její odhady.**
K Turingovu stroji $M$ jsme zavedli jeho *časovou složitost* jako funkci $T_M : \mathbb{N} \to \mathbb{N}$, kde $T_M(n)$ je maximální délka výpočtu stroje $M$ nad vstupem velikosti (tj. délky) $n$.

Např. $T_M(50) = 2560$ znamená, že výpočet $M$ pro alespoň jeden vstup $w$, kde $|w| = 50$, je dlouhý 2560 (jedná se tedy o posloupnost konfigurací $C_0, C_1, C_2, \ldots, C_k$ s $k$ *kroky*, kde $k = 2560$) a délka výpočtu $M$ pro každý vstup $u$, kde $|u| = 50$, je nanejvýš 2560.

Jedná se zde o tzv. *worst-case complexity*; studuje se např. i *average-case complexity*.

Připomněli jsme mj. značení $O, o, \Theta$, které porovnává funkce z hlediska jejich asymptotického růstu:

- $f(n) \in O(g(n))$, jestliže $(\exists c \in \mathbb{N})(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)\ f(n) \leq c \cdot g(n)$;
- $f(n) \in \Theta(g(n))$, jestliže $f(n) \in O(g(n))$ a $g(n) \in O(f(n))$;
- $f(n) \in o(g(n))$, jestliže $(\forall c \in \mathbb{N})(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)\ c \cdot f(n) < g(n)$.

Připomněli jsme si (standardní jednopáskový) Turingův stroj $M_1$, který rozhoduje, zda pro zadané slovo $w \in \{a, b\}^*$ platí $w = w^R$ (tedy zda $w$ je palindromem), a odvodili jsme, že pro časovou složitost $T_{M_1}$ platí $T_{M_1}(n) \in \Theta(n^2)$.

Pak jsme navrhli dvoupáskový stroj $M_2$, který rozhoduje tentýž problém a pro jeho časovou složitost platí $T_{M_2}(n) \in \Theta(n)$.

**Simulace vícepáskového stroje jednopáskovým s nanejvýš kvadratickou ztrátou.**

**Tvrzení 11** *Vícepáskový Turingův stroj $M$ s časovou složitostí $T_M$ lze simulovat jednopáskovým Turingovým strojem $M'$ s časovou složitostí $T_{M'}(n) \in O((T_M(n))^2)$.*

**Důkaz.** (Idea.)

Jakýkoli $k$-páskový stroj $M$ lze simulovat jednopáskovým strojem $M'$ s "$k$-stopou" páskou, kde je v každé stopě poznačena aktuální pozice hlavy simulovaného stroje. Během výpočtu nad $w$ se pozice hlav od sebe vzdálí maximálně o $2 \cdot T_M(|w|)$ políček (proč?). Existuje tedy (malá) konstanta $c_1$ taková, že jeden krok stroje $M$ je simulován nanejvýš $c_1 \cdot T_M(|w|)$ kroky stroje $M'$. Z toho snadno plyne, že $T_{M'}(n) \in O((T_M(n))^2)$. □

**Třídy časové složitosti, Time $(f(n))$, PTIME.**
Pro funkci $f : \mathbb{N} \to \mathbb{N}$ definujeme $\textsc{Time}\,(f(n))$ jako třídu (nebo též množinu) problémů, které jsou řešitelné Turingovými stroji s časovou složitostí v $O(f(n))$. Pak jsme definovali třídu

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

Uvědomili jsme si, že PTIME obsahuje právě ty problémy, které jsou řešitelné Turingovými stroji se složitostí omezenou polynomy (pro takový $M$ máme $T_M(n) \leq pol(n)$ pro nějaký polynom $pol$ jedné proměnné).

**Robustnost třídy PTIME.**
Připomněli jsme si jednoduchý fakt (složení polynomů je polynom):

**Pozorování 12** *Pro polynomy* $pol_1 : \mathbb{N} \to \mathbb{N}$ *a* $pol_2 : \mathbb{N} \to \mathbb{N}$ *je funkce* $f : \mathbb{N} \to \mathbb{N}$ *definovaná* $f(n) = pol_2(pol_1(n))$ *také polynomem (se stupněm* $d = d_1 \cdot d_2$*, kde* $d_1$ *je stupeň polynomu* $pol_1$ *a* $d_2$ *je stupeň polynomu* $pol_2$*).*

Díky Tvrzení 20 a Pozorování 12 vidíme, že obsah třídy PTIME je nezávislý na tom, zda jako referenční model bereme standardní jednopáskové či vícepáskové stroje. Obecněji lze ukázat, že pojem třídy PTIME je *robustní* v tom smyslu, že je nezávislý na zvoleném referenčním modelu, pokud ten je "rozumný" (což v technickém smyslu znamená, že je možné jej simulovat Turingovými stroji s nanejvýš polynomiální ztrátou, a naopak).

Proto lze vcelku volně hovořit o *polynomiálních algoritmech* (což jsou ty, kterým odpovídají Turingovy stroje s polynomiální složitostí, přesněji řečeno se složitostí omezenou polynomem) a *polynomiálních problémech* (což jsou ty, které lze řešit polynomiálními algoritmy).

*Cvičení.*

Mj. jsme se podívali na strukturu ukázky druhé zápočtové písemky a připomněli si uspořádání běžných funkcí z hlediska asymptotického růstu.

## Týden 8

Připomněli jsme si polynomiální algoritmy pro (již studentům známé) problémy; jsou to např.

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

Pak jsme se zamysleli nad problémem vrcholového pokrytí v grafu:

> Vertex Cover (VC)
>
> *Instance (neboli vstup):* Neorientovaný graf $G = (V, E)$ (kde $V$ je množina *vrcholů* a $E$ je množina *hran*, tj. množina obsahující některé (neuspořádané) dvojice vrcholů).
>
> *Výstup:* Nejmenší vrcholové pokrytí v $G$, tedy množina $V' \subseteq V$ s nejmenší možnou mohutností, která splňuje, že každá hrana $e \in E$ má neprázdný průnik s $V'$ (tedy každá hrana je pokryta množinou $V'$, tj. má alespoň jeden konec ve $V'$).

Podobně jako např. u problému minimální kostry se jedná o *optimalizační problém*; výstup také není obecně určen jednoznačně. (Abychom přesně vyhověli naší definici problému, což je trojice $(IN, OUT, f)$, kde $f$ je funkce typu $IN \to OUT$, museli bychom specifikaci problému doplnit.)

Uvedli jsme, že pro problém VC není znám polynomiální algoritmus, ani není dokázáno, že neexistuje. To platí dokonce i pro jednodušší ANO/NE problém, který je tzv. *rozhodovací verzí* (decision version) problému VC:

> $\mathrm{VC}_{dec}$
>
> *Instance:* instance problému VC, tedy graf $G = (V, E)$, a číslo $k$ $(0 \leq k \leq |V|)$.
>
> *Otázka:* Existuje v $G$ vrcholové pokrytí $V'$, kde $|V'| = k$ ?

Všimli jsme si úzkého vztahu problému VC k problému IS (Independent Set), neboli k hledání maximální nezávislé množiny vrcholů v (neorientovaném) grafu. (Instance IS jsou stejné jako u VC, hledáme ale množinu vrcholů s největší mohutností takovou, že žádná hrana nemá oba konce v oné množině.) V problému $\mathrm{IS}_{dec}$ obsahuje instance vedle grafu i číslo $k$ a otázkou je, zda existuje nezávislá množina velikosti $k$.

Mj. pro exaktnější vyjádření vztahu problémů $\mathrm{VC}_{dec}$ a $\mathrm{IS}_{dec}$ jsme zavedli na ANO/NE problémech (či příslušných jazycích) relaci $\leq_{\mathrm{p}}$, která je zjemněním (podmnožinou) dříve zavedené relace $\leq_{\mathrm{m}}$:

Pro ANO/NE problémy $P_1, P_2$ platí

$$P_1 \leq_{\mathrm{p}} P_2,$$

což čteme $P_1$ je *polynomiálně převeditelný*, nebo též *polynomiálně redukovatelný*, *na* $P_2$, jestliže existuje polynomiální algoritmus $A$, který k instanci $I_1$ problému $P_1$ sestrojí instanci $I_2$ (také označenou např. $A(I_1)$) problému $P_2$, přičemž je zaručeno, že odpověď $P_1$ na $I_1$ je stejná jako odpověď $P_2$ na $I_2$.

Pro jazyk $L_1$ v abecedě $\Sigma_1$ a jazyk $L_2$ v abecedě $\Sigma_2$ máme $L_1 \leq_{\mathrm{p}} L_2$, jestliže existuje funkce $f : \Sigma_1^* \to \Sigma_2^*$, která je vyčíslitelná polynomiálním algoritmem a splňuje $\forall w \in \Sigma_1^* : w \in L_1 \Leftrightarrow f(w) \in L_2$.

Uvědomili jsme si, že platí

**Tvrzení 13** $\mathrm{VC}_{dec} \leq_{\mathrm{p}} \mathrm{IS}_{dec}$ *a* $\mathrm{IS}_{dec} \leq_{\mathrm{p}} \mathrm{VC}_{dec}$.

**Důkaz.** Všimněme si, že v grafu $G = (V, E)$ je $V'$ nezávislá množina právě tehdy, když $V \setminus V'$ je vrcholové pokrytí.

Pro prokázání jak $\mathrm{VC}_{dec} \leq_{\mathrm{p}} \mathrm{IS}_{dec}$, tak $\mathrm{IS}_{dec} \leq_{\mathrm{p}} \mathrm{VC}_{dec}$, stačí tedy použít algoritmus, který ke vstupu $G = (V, E), k$ vydá výstup $G = (V, E), |V| - k$. □

Uvědomili jsme si také platnost jednoduchých tvrzení:

**Tvrzení 14** *Relace* $\leq_{\mathrm{p}}$ *je tranzitivní.*

**Tvrzení 15** *Jestliže* $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ *a* $\mathcal{P}_2 \in \mathrm{PTIME}$*, pak* $\mathcal{P}_1 \in \mathrm{PTIME}$.
*(Neboli: Jestliže* $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ *a* $\mathcal{P}_1 \notin \mathrm{PTIME}$*, pak* $\mathcal{P}_2 \notin \mathrm{PTIME}$*.)*

Ve smyslu příslušnosti či nepříslušnosti k PTIME jsou na tom tedy problémy VC a IS (či jejich rozhodovací verze) stejně.

> Je zřejmé, že pokud by existoval polynomiální algoritmus řešící VC, tak existuje polynomiální algoritmus rozhodující $\mathrm{VC}_{dec}$. Umíte odvodit i opačné tvrzení? (Není to těžké, ale chce to trochu více zamyšlení.)

Připomněli jsme problém SAT (Instance: booleovská formule $\varphi$; otázka: je $\varphi$ splnitelná?) a uvedli jsme jeho podproblém 3-SAT (Instance: booleovská formule $\varphi$ v konjunktivní normální formě, v níž každá klauzule obsahuje nejvýš tři literály; otázka: je $\varphi$ splnitelná?)

Ukázali jsme si konstrukce prokazující, že

$$\mathrm{SAT} \leq_{\mathrm{p}} \text{3-SAT} \leq_{\mathrm{p}} \mathrm{IS}_{dec}.$$

(Převod SAT na 3-SAT je podrobně popsán např. v Animacích přístupných z webové stránky předmětu.
Idea převodu 3-SAT na $\mathrm{IS}_{dec}$ je jednoduchá: ke každému výskytu literálu ve vstupní formuli [tj. v instanci problému 3-SAT] vytvoříme vrchol grafu a hranou spojíme vrcholy odpovídající dvěma výskytům literálů pokud jsou tyto výskyty ve stejné klauzuli anebo jsou komplementární, tedy ve tvaru $x$ a $\neg x$ pro nějakou výrokovou proměnnou $x$); doplníme číslo rovnající se počtu klauzulí a máme tak kýženou instanci problému $\mathrm{IS}_{dec}$.)

Díky Tvrzení 14 tak máme i $\mathrm{SAT} \leq_{\mathrm{p}} \mathrm{IS}_{dec}$.

> Umíte ukázat, že také platí $\mathrm{IS}_{dec} \leq_{\mathrm{p}} \mathrm{SAT}$ ?

## Týden 9

Dokončili jsme demonstraci polynomiálních redukcí ve "šňůře"

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

kde

- 3-SAT je podproblém problému SAT: instancí je booleovská formule v 3-CNF (tedy v konjunktivní normální formě, kde v každé klauzuli jsou nejvýš tři literály) a otázkou je, zda je daná formule splnitelná;
- VC je optimalizační problém, v němž se k zadanému neorientovanému grafu hledá minimální vrcholové pokrytí (tj. množina vrcholů, v níž má každá hrana aspoň jeden konec);
- $\text{VC}_{dec}$ je rozhodovací verze problému VC (instancí je graf $G$ a číslo $k$; otázkou jde, zda v $G$ existuje vrcholové pokrytí velikosti $k$);
- HC je problém, v němž se tážeme zda v daném orientovaném grafu existuje hamiltonovský cyklus (tedy cesta procházející každým vrcholem právě jednou a končící ve vrcholu, v němž začíná);
- HK je problém, v němž se tážeme zda v daném neorientovaném grafu existuje hamiltonovská kružnice (tedy cesta procházející každým vrcholem právě jednou a končící ve vrcholu, v němž začíná);
- TSP (Travelling Salesman Problem, problém obchodního cestujícího) je optimalizační problém, jehož instancí je úplný graf s ohodnocenými hranami, v němž se hledá nejkratší okružní cesta;
- $\Delta$-TSP (též "metrický TSP") je podproblém problému TSP, v němž každá instance splňuje trojúhelníkovou nerovnost (tedy $d(i,j) \leq d(i,k) + d(k,j)$, kde $i, j, k$ jsou vrcholy a $d(x,y)$ je hodnota ["délka"] hrany {x,y});
- $\Delta\text{-TSP}_{dec}$ a $\text{TSP}_{dec}$ jsou příslušné rozhodovací verze.

**SAT $\leq_{\text{p}}$ 3-SAT**

Zde jsme si uvědomili, že k obecné booleovské formuli $\varphi$ lze rychle (standardním polynomiálním algoritmem realizujícím např. syntaktickou analýzu výrazů v překladačích programovacích jazyků) sestrojit příslušný logický obvod (boolean circuit), který má na vstupu booleovské proměnné vyskytující se ve formuli a jehož hradla odpovídají výskytům logických spojek ve formuli. Každému hradlu jsme přiřadili novou booleovskou proměnnou a přirozeně jsme sestavili formuli $\varphi'$ v 3-CNF, která popisuje správný výpočet obvodu, jehož výstupem má být hodnota 1. Formule $\varphi'$ je tedy splnitelná právě tehdy, když $\varphi$ je splnitelná.

**$\text{IS}_{dec} \leq_{\text{p}} \text{VC}_{dec}$**

Tady to je jasné díky jednoduchému pozorování:

*Tvrzení.* V grafu $G = (V, E)$ je $V'$ nezávislá množina právě tehdy, když $V \setminus V'$ je vrcholové pokrytí.

**$\mathbf{VC}_{dec} \leq_{\mathrm{p}}$ HC**
To bylo technicky komplikovanější. Pro (neorientovaný) graf $G = (V, E)$ a číslo $k$ jsme sestrojili graf $G' = (V', E')$ postupně takto:

- pro každý vrchol $v \in V$ grafu $G$, se stupněm [tedy počtem s ním incidentních hran] $deg(v)$, zařadíme do $G'$ "řetízek" vrcholů $v_1 \to v_2 \cdots \to v_{2 \cdot deg(v)}$ (šipky označují příslušné hrany v $G'$);
- pro dané číslo $k$ zařadíme do $G'$ navíc vrcholy $u_1, u_2, \ldots, u_k$ a pro každé $i \in [1, k]$ a každý vrchol $v \in V$ s nenulovým stupněm přidáme do $G'$ hrany $u_i \to v_1$ a $v_{2 \cdot deg(v)} \to u_i$ (z $u_i$ lze tedy "skočit" na začátek libovolného "řetízku" a z konce libovolného řetízku lze zase skočit na $u_i$, pro každé $u_i \in \{u_1, u_2, \ldots, u_k\}$);
- pro každý vrchol $v \in V$ grafu $G$ očíslujeme jeho incidentní hrany čísly $1, 2, \ldots, deg(v)$ a pro každou hranu $e = \{v, v'\} \in E$ grafu $G$ přidáme do $G'$ čtyři (orientované) hrany takto: nechť $\ell$ je číslo hrany $e$ vzhledem k $v$ a $\ell'$ je číslo hrany $e$ vzhledem k $v'$; pak přidáme hrany $v_{2\ell-1} \leftrightarrow v'_{2\ell'-1}$ a $v_{2\ell} \leftrightarrow v'_{2\ell'}$ (kde $x \leftrightarrow y$ reprezentuje dvojici hran $x \to y$ a $y \to x$).

Prověřili jsme, že skutečně platí

a) když v $G$ existuje vrcholové pokrytí velikosti $k$, tak v $G'$ existuje hamiltonovský cyklus, a

b) když v $G'$ existuje hamiltonovský cyklus, tak v $G$ existuje vrcholové pokrytí velikosti $k$.

**HC $\leq_{\mathrm{p}}$ HK**
V daném orientovaném grafu $G$ každý vrchol $v$ nahradíme trojicí vrcholů $v_1, v_2, v_3$ s (neorientovanými) hranami $\{v_1, v_2\}$ a $\{v_2, v_3\}$. Pak každou (orientovanou) hranu $(v, v')$ nahradíme neorientovanou hranou $\{v_3, v'_1\}$; tím vznikne neorientovaný graf $G'$. Není těžké ověřit, že v $G$ existuje hamiltonovský cyklus právě tehdy, když v $G'$ existuje hamiltonovská kružnice.

Prokázání HK $\leq_{\mathrm{p}}$ HC je jednodušší. Promyslete si to!

**HK $\leq_{\mathrm{p}}$ $\Delta$-$\mathbf{TSP}_{dec}$**
V grafu $G = (V, E)$ (instanci problému HK) jsme každé hraně přiřadili hodnotu (délku) 1 a pak jsme doplnili hrany tak, aby vznikl úplný graf; každé doplněné hraně jsme přiřadili hodnotu 2. Snadno jsme si ověřili, že ve vzniklém ohodnoceném úplném grafu $G'$ je splněna trojúhelníková nerovnost a v $G$ existuje hamiltonovská kružnice právě tehdy, když v $G'$ existuje okružní cesta (neboli hamiltonovská kružnice) délky $|V|$.

**Rychlé řešení $\mathbf{TSP}_{dec}$ stačí pro rychlé řešení TSP.**
Uvědomili jsme si, že kdybychom měli polynomiální algoritmus rozhodující $\mathrm{TSP}_{dec}$, tak bychom sestrojili i polynomiální algoritmus řešící optimalizační problém TSP.

**Časová složitost nedeterministických strojů, třída** NPTIME**.**
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). $T_M(n)$ je opět délkou nejdelšího výpočtu stroje $M$ nad vstupem velikosti $n$ (pokud jsou všechny takové výpočty konečné).

Zavedli jsme třídy problémů $\textsc{NTime}\,(f(n))$ (obsahující ty problémy, které lze rozhodovat nedeterministickými stroji s časovou složitostí v $O(f(n))$) a definovali jsme

$$\mathrm{NPTIME} = \bigcup_{k \in \mathbb{N}} \textsc{NTime}\,(n^k).$$

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

**Tvrzení 16** $\mathrm{P} \subseteq \mathrm{NP}$.

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

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

**Tvrzení 17** *Jestliže* $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ *a* $\mathcal{P}_2 \in \mathrm{NP}$*, pak* $\mathcal{P}_1 \in \mathrm{NP}$*.*
*(Neboli: Jestliže* $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ *a* $\mathcal{P}_1 \notin \mathrm{NP}$*, pak* $\mathcal{P}_2 \notin \mathrm{NP}$*.)*

**TSP$_{dec}$ patří do** NP**.**
Ukázali jsme, že $\mathrm{TSP}_{dec}$ patří do NP (a tedy mj. všechny problémy ve šnůře (3) jsou v NP). Načrtli jsme prostě pseudokód jednoduchého nedeterministického algoritmu, který $\mathrm{TSP}_{dec}$ rozhoduje a má očividně polynomiální složitost:

1. pro daný ohodnocený úplný graf a limit $\ell$ postupně (nedeterministicky) zvol permutaci množiny vrcholů (když množina vrcholů je $\{1, 2, \ldots, n\}$, tak pro $j = 1, 2, \ldots, n$ postupně vyber $i_j \in \{1, 2, \ldots, n\} \setminus \{i_1, i_2, \ldots, i_{j-1}\}$);
2. spočti délku $d$ okružní cesty určené zvolenou permutací;
3. pokud $d \leq \ell$, vrať ANO, jinak vrať NE.

NP**-těžké a** NP**-úplné problémy.**
Uvedli jsme důležitou definici:

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

Mj. jsme si všimli:

**Tvrzení 18** *Jestliže* $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ *a* $\mathcal{P}_1$ *je* NP*-těžký, pak* $\mathcal{P}_2$ *je* NP*-těžký.*

Naznačili jsme si důkaz následující důležité věty, díky níž jsme pak mj. vyvodili, že všechny problémy ve šnůře (3) jsou NP-úplné.

**Věta 19** *(Cook) Problém SAT je NP-úplný.*

**Důkaz.** Už víme, že SAT je v NP (navíc snadno umíme navrhnout přímý nedeterministický algoritmus prokazující tento fakt).

Idea důkazu faktu, že SAT je NP-těžký:

Uvažme problém $\mathcal{P}$ z NP; problém $\mathcal{P}$ je tedy rozhodován nedeterministickým Turingovým strojem $M$ s časovou složitostí $T_M(n) \in O(n^k)$ pro nějaké $k \in \mathbb{N}$. Výpočet stroje $M$ nad slovem $w$ velikosti $n$ chápeme jako posloupnost $O(n^k)$ konfigurací velikosti $O(n^k)$ zapsaných "pod sebou". Tak vznikne dvourozměrná tabulka velikost $O(n^{2k})$. Pomocí booleovské formule, která je konjunkcí malých "lokálních" podmínek, popíšeme, kdy se vyplnění oné tabulky skutečně rovná akceptujícímu výpočtu stroje $M$ na slově $w$. Takto je vidět, že problém $\mathcal{P}$ lze polynomiálně převést na SAT (tedy platí $\mathcal{P} \leq_{\mathrm{p}}$ SAT). $\square$

Uvedli jsme i třídu co-NP a diskutovali NP $\cap$ co-NP.

Také jsme zmínili nejasný status *problému grafového izomorfismu*: je očividně v NP, ale neumíme ukázat, že by patřil do P, ani neumíme ukázat, že je NP-těžký.

## 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".

## Týden 10

**Konkrétní problém v** PSPACE**.**
Uvedli jsme "oblázkovou hru":

> Problém $\textsc{MinPebble}_{dec}$
>
> 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 $\textsc{MinPebble}_{dec}$ 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 20** *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, z níž plyne PSPACE = NPSPACE.

**Savitchova věta.**

**Věta 21** *(Savitch) Pro každou funkci $f : \mathbb{N} \to \mathbb{N}$ splňující $f(n) \geq n$ platí*

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

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).

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

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

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

Odvodili jsme mj.:

**Tvrzení 22** *Jestliže $\mathcal{P}_1 \leq_{\mathrm{p}} \mathcal{P}_2$ a $\mathcal{P}_1$ je* PSPACE-*těžký, pak $\mathcal{P}_2$ je* PSPACE-*těžký.*

Uvedli jsme analogii Cookovy věty pro PSPACE; místo SAT uvažujeme problém QBF (někdy označovaný i Q-SAT), který se ptá, zda daná *plně kvantifikovaná* booleovská formule je pravdivá.

**Věta 23** *Problém QBF je* PSPACE-*úplný.*

**QBF je v** PSPACE**.**
Příslušnost QBF k PSPACE prokážeme přímočarým rekurzivním algoritmem. Návod: Předpokládáme zadanou formuli v prenexní formě, tedy nejdříve jsou uvedeny kvantifikátory a na konci je (pod)formule bez kvantifikátorů; např. $(\forall x_1)(\exists x_2)(\forall x_3)\,(x_1 \vee (x_2 \to x_3))$ je v prenexní formě. Popište, jak rozhodnete (ne)pravdivost takové formule $(Q_1x_1)(Q_2x_2)\cdots(Q_nx_n)\varphi$ pomocí sestrojení binárního stromu hloubky $n$, jehož větve odpovídají všem pravdivostním ohdnocením výrokových proměnných $x_1, x_2, \ldots, x_n$. Pak si uvědomte, že příslušné vyhodnocení (končící rozhodnutím, zda kořen stromu je "pravdivý") lze provést průchodem do hloubky, při němž si v paměti udržujeme jen aktuální větev stromu (nikoli celý strom).

**QBF je** PSPACE**-těžký.** (Toto není potřeba ke zkoušce.)
PSPACE-obtížnost problému QBF 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

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

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.

Také jsme uvedli, 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.

Pak se psala druhá zápočtová písemka.

## Zápočtový týden

Program je na webové stránce předmětu.

**Seznam otázek k ústní zkoušce (poznámky k průběhu jsou na další straně).**

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 (Postova věta).
5. Nerozhodnutelné problémy. Důkazy diagonalizační metodou.
6. 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.
7. Důkaz nerozhodnutelnosti ekvivalence bezkontextových gramatik.
8. Riceova věta; její důkaz a aplikace.

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, třídění, ...).
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. PSPACE-úplnost. Příklady PSPACE-úplných problémů. Prokázaní, že problém QBF (pravdivost plně kvantifikovaných booleovských formulí) je v PSPACE.

Poznámky k průběhu zkoušky:
- bez splněného zápočtu nelze jít na zkoušku;
- přihlašování na termíny vypsané v IS STAG, pokud není dohodnuto jinak;
- zkoušení bude typicky probíhat ve zkouškovém období ZS;
- zkouška je ústní, s písemnou přípravou;
- student si vytáhne dvě otázky, jedna bude z okruhu A), druhá z okruhu B);
- prověřuje se mj. schopnost demonstrace pojmů a tvrzení konkrétními příklady ...;
- čas na písemnou přípravu (jen s papírem a tužkou, bez dalších materiálů): $\geq$ 15 minut;
- čas na ústní zkoušení (u tabule): zhruba 20 minut či více (podle situace).