Arora, Barak: Computational complexity: A modern approach.

Definice. Zavedeme si složitostní třídy:

- $\mathsf{L} = \mathsf{LOG} = \mathrm{DSPACE}(\log n)$
- $\mathsf{NL} = \mathsf{NLOG} = \mathrm{NSPACE}(\log n)$
- $\mathsf{POLYLOG} = \bigcup_i \mathrm{NSPACE}(\log^i n)$
- $\mathsf{PSPACE} = \bigcup_i \mathrm{DSPACE}(n^i)$
- $\mathsf{EXPSPACE} = \bigcup_c \mathrm{DSPACE}(2^{cn})$
- $\mathsf{P} = \bigcup_i \mathrm{DTIME}(n^i)$
- $\mathsf{NP} = \bigcup_i \mathrm{NTIME}(n^i)$
- $\mathsf{EXT} = \bigcup_c \mathrm{DTIME}(2^{cn})$
- $\mathsf{NEXT} = \bigcup_c \mathrm{NTIME}(2^{cn})$
- $\mathsf{EXPTIME} = \bigcup_c \mathrm{DTIME}(2^{n^c})$
- $\mathsf{NEXPTIME} = \bigcup_c \mathrm{NTIME}(2^{n^c})$

Všimněme si, že zatímco víme pouze $\mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$, alespoň jedna nerovnost je ostrá, jelikož $\mathsf{NL} \subsetneq \mathsf{PSPACE}$.

1 Nedeterministcká složitost

Dříve jsme si ukázali, jak pro složitostní třídy dokazovat jejich vzájemné inkluze. Obecně máme mnoho způsobů, jak porovnat neostrou inkluzí.

Kromě toho máme věty o hiearchii, jenž umožňují o dvou třídach složitosti říci, že se nerovnají. Avšak zatím umíme jen deterministické verze pro prostor a čas. Dále známe Savičovu větu, která nám říká inkluzi nedeterministické prostorové třídy v deterministické.

Kombinací těchto vět můžeme dojít k dalším způsobům, jak oddělovat třídy složitosti, tentokrát i pro nedeterministické třídy.

Lemma 1.1. *Nechť $s_1(n), s_2(n), f(n)$ jsou prostorově konstruovatelné funkce takové, že $s_2(n) \geq n, f(n) \geq n$. Potom* $\mathrm{NSPACE}(s_1(n)) \subseteq \mathrm{NSPACE}(s_2(n)) \Rightarrow \mathrm{NSPACE}(s_1(f(n))) \subseteq \mathrm{NSPACE}(s_2(f(n)))$.

Důkaz. Vezměme $L_1 \in \text{NSPACE}(s_1(f(n)))$, ten má NTS M_1 rozpoznávající jazyk L_1 v prostoru $s_1(f(n))$. Definujme jazyk L_2 jako řetězce ve tvaru $x\$^i$ takové, že je NTS M_1 přijímá v prostoru $s_1(|x| + i)$. Myšlenka je taková, že uměle nafukujeme vstup tak, aby stačilo z dlouhého vstupu přečíst krátký kus. Pro $i = f(|x|) - |x|$ platí, že $x\$^i \in L_2 \Leftrightarrow x \in L_1$.

Platí $L_2 \in \text{NSPACE}(s_1(n))$, protože NTS M_2 pracuje v prostoru $s_1(n)$ tak, že na vstupu $x\$^i$ simuluje NTS M_1 na vstupu x. Z předpokladu lemmatu platí, že $L_2 \in \text{NSPACE}(s_2(n))$. To znamená, že existuje NTS M_3, který rozhoduje jazyk L_2 v prostoru $s_2(n)$.

Postavíme NTS M_4 pracující na vstupu x takto:

1. Na první pracovní pásce vyznačí prostor $f(|x|)$.
2. Na druhé pracovní pásce vyznačí prostor $s_2(f(|x|)) \geq f(|x|)$, protože $f(n), s_2(n) \geq n$.
3. Na první pracovní pásku přepíše vstup x.
4. Na vstupu x simuluje chod NTS M_3 na vstupu $x\$^i$, kde $i = f(|x|) - |x|$. Pro účely simulace bude jako vstupní pásku považovat první pracovní pásku a jako pracovní pásku bude považovat druhou pracovní pásku. Pokud hlava simulovaného stroje M_3 má přejet nad suffix $\$^i$, pak si tuto skutečnost budeme značit na první pracovní pásce.

Tím pádem $s_2(|x\$^i|) = s_2(|x|+f(|x|)-|x|) = s_2(f(|x|))$ a stroj M_4 rozpoznává jazyk L_1 v prostoru $s_2(f(|x|))$. □

K čemu se nám toto lemma hodí?

Příklad. Chceme ukázat, že $\text{NSPACE}(n^3) \subsetneq \text{NSPACE}\,(n^4)$. Tyto třídy jsou nedeterministické, tudíž nemůžeme použít standardní větu o hierarchii. Musíme tedy postupovat jinak.

Půjdeme to sporem. Předpokládejme, že $\text{NSPACE}(n^4) \subseteq \text{NSPACE}(n^3)$. Pro předchozí lemma určíme $s_1(n) = n^4, s_2(n) = n^3$.

Použijme nejprve $f(n) = n^3$. Pak z lemmatu dostáváme, že $\text{NSPACE}(n^{12}) \subseteq \text{NSPACE}(n^9)$.

Pokračujme dále. Nyní nechť $f(n) := n^4$, lemma nám říká ($\text{NSPACE}(n^{15}) \subseteq$) $\text{NSPACE}(n^{16}) \subseteq \text{NSPACE}(n^{12})$.

Nakonec použijme $f(n) := n^5$, získáváme $\text{NSPACE}(n^{20}) \subseteq \text{NSPACE}(n^{15})$. Dohromady dostáváme, že $\text{NSPACE}(n^{20}) \subseteq \text{NSPACE}(n^9)$.

Nyní máme dostatečně velký polynom, abychom mohli použít Savičovu větu v kombinaci s větou o deterministické prostorové hierarchii: $\text{NSPACE}(n^9) \subseteq$

$\mathrm{DSPACE}(n^{18}) \subsetneq \mathrm{DSPACE}(n^{20}) \subseteq \mathrm{NSPACE}(n^{20}) \subseteq \mathrm{NSPACE}(n^9)$. Dostali jsme se ke sporu, jelikož $\mathrm{NSPACE}(n^9) \neq \mathrm{NSPACE}(n^9)$ zjevně nemůže platit.

Tento postup nyní zobecníme pro obecné polynomy:

Věta 1.2. *Nechť* $\varepsilon > 0, r \geq 1$. *Pak* $\mathrm{NSPACE}(n^r) \subsetneq \mathrm{NSPACE}(n^{r+\varepsilon})$.

Důkaz. Pro každé r a ε najdeme $s, t \in \mathbb{N}$ takové, že $r \leq \frac{s}{t} \leq \frac{s+1}{t} \leq r + \varepsilon$. Stačí použít trochu analýzy. Pak stačí ukázat, že $\mathrm{NSPACE}(n^{s/t}) \subsetneq \mathrm{NSPACE}(n^{(s+1)/t})$. Půjdeme na to sporem.

Předpokládejme tedy, že $\mathrm{NSPACE}(n^{(s+1/t)} \subseteq \mathrm{NSPACE}(n^{s/t})$. To je navíc naše s_1 a s_2. Nechť $f(n) := n^{(s+i)t}$ pro $i \in \{0, \ldots, s\}$.

Pro $i = 0$ dostáváme $\mathrm{NSPACE}(n^{s(s+1)}) \subseteq \mathrm{NSPACE}(n^{s^2})$, zbavili jsme se tedy exponentu.

Pokud pro nějaké i platí $\mathrm{NSPACE}(n^{(s+i)(s+1)}) \subseteq \mathrm{NSPACE}(n^{(s+i)s})$, pak pro $i + 1$ díky tranzitivitě platí $\mathrm{NSPACE}(n^{(s+i+1)(s+1)}) \subseteq \mathrm{NSPACE}(n^{(s+i+1)s})$.

Zvolme tedy $i = s$, dostáváme $\mathrm{NSPACE}(n^{2s(s+1)}) \subseteq \mathrm{NSPACE}(n^{2s^2})$. Abychom mohli pokračovat, musí platit $(s + i + 1)s \leq (s + i)(s + 1)$, což si čtenář jistě sám ověří.

Zřetězením inkluzí dospějeme ke vztahu $\mathrm{NSPACE}(n^{2s(s+1)}) \subseteq \mathrm{NSPACE}(n^{s^2})$. V tuto chvíli přijde na řadu Savičova věta: $\mathrm{NSPACE}(n^{s^2}) \subseteq \mathrm{DSPACE}(n^{2s^2})$. Dále pak použijeme větu o prostorové hierarchii a získaný vztah:
$\mathrm{DSPACE}(n^{2s^2}) \subsetneq \mathrm{DSPACE}(n^{2s(s+1)}) \subseteq \mathrm{NSPACE}(n^{2s(s+1)}) \subseteq \mathrm{NSPACE}(n^{s^2})$.
Samozřejmě, protože množina je rovna sama sobě, máme hledaný spor. □

2 Polynomiální hieararchie

Definice (DTS s orákulem)**.** DTS s orákulem A, kde A je jazyk, je standardní DTS rozšířené následujícím způsobem:

- Má navíc dotazovací pracovní pásku, používající abecedu jazyka A,
- má navíc 3 stavy: DOTAZ, ANO, NE
- Pokud během práce DTS přejde do stavu DOTAZ, tak v následujícím kroku přejde do stavu ANO, pokud aktuální obsah dotazové pásky je slovo z A, jinak NE. Navíc se po každém dotazu páska vyprázdní.

Jazyk slov přijímaných DTS M s orákulem A značíme $L(M, A)$.

Úplně stejným způsobem jde rozšířit o orákulum NTS. DTS s (dobrým) orákulem jsou alespoň tak silné, jako NTS. Navíc můžeme orákulu dát nerozhodnutelný jazyk. Síla orákula závisí na dodaném jazyku.

Definice. Nechť L_1, L_2 jsou jazyky. Řekneme, že L_1 je Turingovsky (deterministicky) převoditelný na L_2 ($L_1 \leq_T L_2$) v polynomiálním čase, pokud existuje DTS M s orákulem pracující v polynomiálním čase takový, že $L_1 = L(M, L_2)$.

Definice. Nechť A je jazyk, pak $P(A) = \{B \mid B \leq_T A\}$. Když $\mathcal{C}$ je třída jazyků, pak $P(\mathcal{C}) = \{B \mid \exists A \in \mathcal{C} : B \leq_T A\}$.

Podívejme se na $\mathsf{P} = \bigcup_c \mathrm{DTIME}(n^c)$. Potom $P(\mathsf{P}) = \mathsf{P}$, nijak jsme si nempomohli:

Jistě vidíme, že $\mathsf{P} \subseteq P(\mathsf{P})$. Podle definice $B \in P(\mathsf{P})$ právě, když $\exists A \in \mathsf{P} : B = L(M, A)$ pro nějaký DTS M pracující v polynomiálním čase. Protože $A \in \mathsf{P}$, pak $\exists M' : A = L(M')$, kde M' pracuje v polynomiálním čase.

Zkonstruujeme tudíž $\tilde{M}$ přijímající B, který pracuje jako M a vždy, když M přijde do stavu DOTAZ, tak $\tilde{M}$ pustí M' na slově z dotazovací pásky. Máme nejvýše polynomiálně mnoho dotazů, které trvají polynomiálně dlouho, tudíž i $\tilde{M}$ pracuje v polynomiálním čase a $B \in \mathsf{P}$.

Přejděme k nedeterministickým TS. Definice Turingovské převoditelnosti se nám trochu změní:

Definice. Nechť L_1, L_2 jsou jazyky. Řekneme, že L_1 je nedeterministicky Turingovsky převoditelný na L_2 ($L_1 \leq_{NP} L_2$) v polynomiálním čase, pokud existuje NTS M s orákulem pracující v polynomiálním čase takový, že $L_1 = L(M, L_2)$.

Definice. Nechť A je jazyk, pak $NP(A) = \{B \mid B \leq_{NP} A\}$. Když $\mathcal{C}$ je třída jazyků, pak $NP(\mathcal{C}) = \{B \mid \exists A \in \mathcal{C} : B \leq_{NP} A\}$.

Tuto definici můžeme rozšířit na libovolnou složitostní třídu a takovým třídám se říkají *relativizované*.

Pojďme si romyslet, co je tentokrát $NP(\mathsf{P})$ a $NP(\mathsf{NP})$. Očividně $NP(\mathsf{P}) = \mathsf{NP}$ z podobného důvodu, jako u $P(\mathsf{P})$. Pro druhou třídu se dostaneme do problémů. Pokud se pokusíme odsimulovat výpočet v NTS, tak „nalepením“ výpočtu orákula můžeme omylem vytvořit nové přijímající stavy. Nemusí to proto být NP, ale něco potenciálně většího. Zatím nemůžeme určit, zda ostře většího.

Lemma 2.1. *Pro každý jazyk A platí $P(A) \subseteq NP(A) \subseteq PSPACE(A)$.*

Nyní máme všechny prostředky k tomu, abychom zavedli polynomiální hiearchii.

Definice. Definujeme třídy jazyků $\Sigma_0, \Sigma_1, \Sigma_2, \ldots$, kde $\Sigma_0 = \mathsf{P}$ a $\forall k \geq 0 : \Sigma_{k+1} = NP(\Sigma_k)$.

Potom polynomiální hiearchií rozumíme třídu $\mathsf{PH} = \bigcup_k \Sigma_k$.

Věta 2.2. $\mathsf{PH} \subseteq \mathsf{PSPACE}$.

Důkaz. Indukcí podle i dokážeme $\Sigma_i \subseteq \mathsf{PSPACE}$:

1. $i = 0$, pak $\Sigma_0 = \mathsf{P} \subseteq \mathsf{PSPACE}$.
2. Předpokládejme, že podmínka platí pro všechna $0 \leq i' \leq i$. Potom $\Sigma_{i+1} \subseteq NP(\Sigma_i) \subseteq PSPACE(\mathsf{PSPACE})$.

   Nechť $B \in PSPACE(\mathsf{PSPACE})$, tedy $\exists A \in \mathsf{PSPACE}$ a DTS M pracující v polynomiálním prostoru. Taktéž existuje DTS M' pracující v poly prostoru přijímající A. Zkonstruujeme $\tilde{M}$ simulující M a vždy, když přejde na DOTAZ, spustí M', aby jej zodpověděl.

   Dotazovací páska se počítá do pracovního prostoru, tudíž DOTAZ je polynomiálně dlouhý, tudíž i M' je poly. Protože pásku mezi dotazy vždy vyprázdníme, můžeme ji znova použít. Tudíž $B \in \mathsf{PSPACE}$. Proto $\Sigma_{i+1} \subseteq \mathsf{PSPACE}$. □

Nyní si však polynomiální hiearchii nadefinujeme pořádněji:

Definice. Polynomiální hiearchie je strukturu tvořena třídami Σ_k, Π_k a Δ_k pro $k \geq 0$, kde:

- $\Sigma_0 = \Pi_0 = \Delta_0 = \mathsf{P}$,
- $\Sigma_{k+1} = NP(\Sigma_k)$,
- $\Pi_{k+1} = Co\text{-}NP(\Sigma_k)$,
- $\Delta_{k+1} = P(\Sigma_k)$,
- $\mathsf{PH} = \bigcup_k \Sigma_k$.

Pokud máme orákulum z jazyka A, potom orákulum z jazyka $\overline{A}$ je stejně silné – můžeme znegovat výsledek orákula. Tudíž $\Sigma_{k+1} = NP(\Pi_k)$ a dále.

Taktéž platí, že $\Sigma_k \cup \Pi_k \subseteq \Delta_{k+1}$ a $\Delta_k \subseteq \Sigma_k \cap \Pi_k$.

Věta 2.3. *Pokud* $\Sigma_k \neq \Pi_k$, *pak* $\Delta_{k+1} \supsetneq \Sigma_k \cup \Pi_k$.

Důkaz. Ukážeme pro $k = 1$. Vezměme jazyk $A = \{(F_1, F_2) \mid F_1 \in \mathrm{SAT}, F_2 \in \overline{\mathrm{SAT}}\}$.

$A \in \Delta_2$: Sestrojíme DTS M s orákulem SAT. Pak se dvěma dotazy zeptáme, zda $F_1 \in SAT$ a $F_2 \notin$ SAT. Pokud ano, přijmeme. Tento DTS je jistě polynomiální.

$A \notin$ Co-NP: Nechť $A' = \{(F, 0) \mid F \in \text{SAT}\} \subseteq A$. Ten je NP-úplný a pro spor předpokládejme, že je i v Co-NP. Pak ale NP = Co-NP. □

Nyní si nadefinujeme polynomiální hiearchii jiným způsobem, a to pomocí omezených kvantifikátorů.

Definice.

- $\exists^{p(n)} x : R(x)$ znamená $\exists x : (|x| \leq p(n)) \wedge (x$ má vlastnost $R)$.
- $\forall^{p(n)} x : R(x)$ znamená $\forall x : (|x| \leq p(n)) \Rightarrow (x$ má vlastnost $R)$.

Definice. Nechť $\mathcal{C}$ je třída jazyků. Pak $\exists\mathcal{C}$ je třída jazyků, kde $A \in \exists\mathcal{C}$ pokud platí $\exists B \in \mathcal{C}\, \exists \text{poly}\, p : x \in A \Leftrightarrow \exists^{p(|x|)} y : \langle x, y\rangle \in B$.

Lemma 2.4. $\exists\mathsf{P} = \mathsf{NP}$.

Důkaz. Nejprve ukážeme $\exists\mathsf{P} \subseteq \mathsf{NP}$. Podle definice $A \in \exists\mathsf{P} \Rightarrow \exists B \in \mathsf{P}\, \exists p : x \in A \Leftrightarrow \exists^{p(|x|)} y : \langle x, y\rangle \in B$.

Mějme NTS M pracující takto: Nejprve M přečte x, následně nedeterministicky uhodne y velikosti nejvýše $p(|x|)$. Nakonec M ověří, zda $\langle x, y\rangle \in B$. Tento stroj jistě pracuje v polynomiálním čase a $A = L(M)$, proto $A \in \mathsf{NP}$.

Naopak $A \in \mathsf{NP}$. Pak existuje NTS M rozpoznávající A v poly čase – $A = L(M)$. Slovo $x \in A$ právě když existuje polynomiálně velké y, které je kódem přijímajícího výpočtu M na vstupu x. Tudíž $x \in A \Leftrightarrow \exists^{p(|x|)} y : \langle x, y\rangle \in B$, kde $B \in \mathsf{P}$. □

Definice. Nechť $\mathcal{C}$ je třída jazyků. Pak $\forall\mathcal{C}$ je třída jazyků, kde $A \in \forall\mathcal{C}$ pokud platí $\exists B \in \mathcal{C}\, \exists \text{poly}\, p : x \in A \Leftrightarrow \forall^{p(|x|)} y : \langle x, y\rangle \in B$.

Lemma 2.5. *Nechť $\mathcal{C}$ je třída jazyků. Pak $A \in \exists\mathcal{C}$ právě, když $\overline{A} \in \forall(Co{-}\mathcal{C})$. Jinými slovy $Co - \exists\mathcal{C} = \forall(Co - \mathcal{C})$.*

Důkaz. Jazyk $A \in \exists\mathcal{C}$ právě, když $\exists B \in \mathcal{C}\, \exists p : x \in A \Leftrightarrow \exists^{p(|x|)} y : \langle x, y\rangle \in B$.
Negací dostáváme $x \in \overline{A} \Leftrightarrow x \notin A \Leftrightarrow \forall^{p(|x|)} y : \langle x, y\rangle \in \overline{B}$, kde $\overline{B} \in Co{-}\mathcal{C}$. □

Z toho dostáváme, že $\forall\mathsf{P} = \forall(Co - \mathsf{P}) = Co - \exists\mathsf{P} =$ Co-NP.

Definice. Třída $\mathcal{C}$ je uzavřená na zdvojování, pokud $\forall A \in \mathcal{C}$ platí, že $B = \{\langle x, y\rangle \mid x \in A\} \in \mathcal{C}$.

Lemma 2.6. *Pokud je $\mathcal{C}$ uzavřená na zdvojování, tak $\mathcal{C} \supseteq \exists\mathcal{C}$ a $\mathcal{C} \supseteq \forall\mathcal{C}$.*

Důkaz. Podle definice $A \in \exists\mathcal{C}$ právě, když $\exists B \in \mathcal{C}$, a tak dále... Zvolme pak $B = A$, pak díky uzavřenosti na zdvojování můžeme ignorovat y. □

Lemma 2.7. *Nechť $\mathcal{C}$ je libovolná třída jazyků z* PH. *Pak $\forall A : A \in \mathcal{C} \Leftrightarrow A^* \in \mathcal{C}$. Jazyk A^* je kódovanou konečnou iterací jazyka A.*

Důkaz. Jistě $A^* \in \mathcal{C}$ implikuje $A \in \mathcal{C}$, protože $A \subseteq A^*$. Předpokládejme nyní, že $A \in \mathcal{C}$ a dokažme $A^* \in \mathcal{C}$.

Nejprve dokažme pro $\mathcal{C} = \mathsf{P} = \Sigma_0 = \Pi_0 = \Delta_0$. Pak prostě spustíme polynomiální DTS na jednotlivá podslova a přijmeme, pokud jsme vždy přijali. Třídy $\mathcal{C} = \Delta_k, k \geq 0$ dokážeme stejným způsobem, máme stále DTS (ale s orákulem).

Nyní mějme $\mathcal{C} = \Sigma_k, k \geq 0$. Tedy $A \in \Sigma_k = NP(\Sigma_{k-1})$. Máme NTS M a $B \in \Sigma_{k-1}$ takové, že $A = L(M, B)$. Zkonstruujeme NTS M^* pro A^* s orákulem B.

Máme vstup $x = (y_1, \ldots, y_n)$. Spustíme M na y_1. Pokud NTS přijme, pokračujeme spuštěním M na y_2, atd. Tím dostaneme strom výpočtu s n patry, která jsou propojená pouze přijímajícím stavem. Všimněme si, že cesta v tomto stromu existuje právě, když v každém patře přijmeme.

Zbývá $\mathcal{C} = \Pi_k, k \geq 0$. Pak $A \in \Pi_k$ právě, když $\overline{Co} - A \in \Sigma_k$. Tudíž existuje NTS M a jazyk $B \in \Sigma_{k-1}$, že $Co - A = L(M, B)$.

Tentokrát však $x \in Co - A^*$, pokud $\exists i : y_i \in Co - A$. Sestavíme NTS M^* takový, že na začátku si nederministicky vybere jedno slovo. Pokud jedno z nich přijme, vyhráli jsme. Tudíž $\overline{Co} - A^* \in \Sigma_k$ a nakonec $A^* \in \Pi_k$. □

To nám říká zajímavou věc. Nechť $A_1, \ldots, A_j \in \Sigma_k$. Pak jazyk $A = \{x \mid \exists y_1 \in A_1, \ldots, y_j \in A_j : x = (y_1, \ldots, y_j)\} \in \Sigma_k$.

Nyní nám stačí jen poslední krok, abychom našli alternativní definici PH.

Věta 2.8.

- $\exists\mathsf{P} = \mathsf{NP}$
- $\forall\mathsf{P} = \mathsf{Co\text{-}NP}$
- $\exists\Sigma_k = \Sigma_k, k > 0$
- $\forall\Pi_k = \Pi_k, k > 0$
- $\exists\Pi_k = \Sigma_{k+1}, k \geq 0$
- $\forall\Sigma_k = \Pi_{k+1}, k \geq 0$

Důkaz. První dvě jsme již ukázali. Dokážeme nyní třetí, ze které vyplyne i čtvrtá.

Už víme, že $\Sigma_k \subseteq \exists\Sigma_k$. Zbývá ukázat $\exists\Sigma_k \subseteq \Sigma_k$. Nechť $A \in \exists\Sigma_k$. To znamená, že $\exists B \in \Sigma_k\, \exists p : x \in A \Leftrightarrow \exists^{p(|x|)} y : \langle x, y\rangle \in B$.

Jazyk $B \in NP(\Sigma_{k-1})$, tudíž máme NTS M a $C \in \Sigma_{k-1}$ takové, že $B = L(M, C)$. Zkonstruujeme NTS M' s orákulem C pracující na vstupu x: Uhádne y polynomiálně velké k x. Pak simuluje chod M na vstupu $\langle x, y\rangle$ a rozhodne, zda patří do B. Pokud ano, $x \in A$.

Tudíž $A = L(M', C)$, proto $A \in NP(\Sigma_{k-1}) = \Sigma_k$. Nyní $\forall\Pi_k = \forall(Co - \Sigma_k) = Co - \exists\Sigma_k = Co - \Sigma_k = \Pi_k$.

Vrhněme se na pátou. Jejím dokázáním symetricky dostaneme i šestou. Ukážeme, že $\exists\Pi_k \subseteq \Sigma_{k+1}$. Pak jazyk $A \in \exists\Pi_k$ právě, když $\exists B \in \Pi_k\, \exists p : x \in A \Leftrightarrow \exists^{p(|x|)} y : \langle x, y\rangle \in B$. Sestrojíme NTS, který uhádne y a vyzkouší $\langle x, y\rangle \in B$.

Pro $\Sigma_{k+1} \subseteq \exists\Pi_k$ ukážeme indukcí podle k. Pro $k = 0$ dostáváme $\Sigma_1 \subseteq \exists\Pi_0 = \exists\mathsf{P} = \mathsf{NP}$.

Platí inkluze $\Sigma_k \subseteq \exists\Pi_{k-1}$ dle IP. Jestliže $A \in \Sigma_{k+1}$, pak existuje NTS M a jazyk B, že $A = L(M, B)$. Slovo $x \in A$ právě, když existuje přijímající výpočet M, který se ptá na slova $w_1, \ldots, w_\ell \in B$, a $z_1, \ldots, z_r$ které patří do $Co - B$.

Jinými slovy $x \in A$ právě, když $\exists^{p(|x|)} y\, \exists^{p(|x|)} w\, \exists^{p(|x|)} z : \langle x, y\rangle \in L, w \in B^*, z \in Co - B^*$, kde $L \in \mathsf{P} \subseteq \Pi_k$. Dále jazyky $Co - B, Co - B^* \in \Pi_k$. Použijeme IP a $B^* \in \Sigma_k \subseteq \exists\Pi_{k-1}$.

Jazyk $B^* \in \exists\Pi_{k-1}$ právě, když existuje $D \in \Pi_{k-1}\, \exists p' : w \in B^* \Leftrightarrow \exists^{p'(|w|)} t : \langle w, t\rangle \in D \in \Pi_k$. Tudíž $x \in A$ právě, když $\exists^{p''(|x|)} y, w, z, t : \langle x, y\rangle \in L, \langle w, t\rangle \in D, z \in Co - B^*$. To taktéž znamená, že $(x, y, w, t, z) \in F \in \Pi_k$, a proto $A \in \exists\Pi_k$. □

Důsledek 2.9 (Definice PH alternujícími kvantifikátory)**.** *Jazyk $A \in \Sigma_k$ právě, když $\exists B \in \mathsf{P}\, \exists p : x \in A \Leftrightarrow \exists^{p(|x|)} y_1 \forall^{p(|x|)} y_2 \cdots : (x, y_1, \ldots, y_k) \in B$.*

Jazyk $A \in \Pi_k$ právě, když $\exists B \in \mathsf{P}\, \exists p : x \in A \Leftrightarrow \forall^{p(|x|)} y_1 \exists^{p(|x|)} y_2 \cdots : (x, y_1, \ldots, y_k) \in B$.

Důkaz. Indukcí podle k. Pro $k = 0$ triviálně platí, zvolíme $A = B \in \mathsf{P}$.

Nyní použijeme indukční krok (věta platí pro $0, 1, \ldots, k$). Uvažme jazyk $A \in \Sigma_{k+1} = \exists\Pi_k$. To dle definice jest právě, když $\exists B \in \Pi_k\, \exists p : x \in A \Leftrightarrow \exists^{p(|x|)} y : (x, y) \in B$.

Dle indukčního předpokladu $B \in \Pi_k$ právě, když $\exists D \in \mathsf{P}\, \exists p' : x \in B \Leftrightarrow \forall^{p'(|x|)} y_1 \exists^{p'(|x|)} y_2 \ldots (x, y_1, \ldots, y_k) \in D$. Můžeme zkombinovat, existuje polynom p'', že $x \in A \Leftrightarrow \exists^{p''(|x|)} \forall^{p''(|x|)} \ldots (x, y, y_1, \ldots, y_k) \in D$. Hotovo. □

Tato definice nám přináší nový výsledek, a to, že polynomiální hiearchie zkolabuje, pokud $\Sigma_k = \Pi_k$:

Důsledek 2.10. *Pokud* $\exists k > 0 : \Sigma_k = \Pi_k$*, pak* $\forall j \geq k : \Sigma_j = \Pi_j = \Sigma_k$.

Důkaz. Indukcí dle j, pro $j = k$ platí triviálně. Pro $j + 1$ ukážeme $\Sigma_{j+1} = \exists\Pi_j = \exists\Sigma_j = \Sigma_j = \Sigma_k$, protože dle IP $\Sigma_{j+1} = \exists\Pi_j$. □

To, že nám polynomiální hiearchie zkolabuje, ještě nutně neznamená, že $\mathsf{P} = \mathsf{NP}$.

Důsledek 2.11. *Buď* $\forall k \geq 0$ *platí* $\Sigma_k \subsetneq \Sigma_{k+1}$ *nebo se* PH *skládá z konečně mnoha tříd* Σ_k.

Důkaz. Nechť existuje k takové, že $\Sigma_k = \Sigma_{k+1}$. Protože víme, že $\Sigma_k \cup \Pi_k \subseteq \Delta_{k+1} \subseteq \Sigma_{k+1}$, platí $\Pi_k \subseteq \Sigma_k$, a proto $\Pi_k = \Sigma_k$. Tudíž PH zkolabuje. □

Důsledek 2.12. *Pokud existuje* k *takové, že* $\mathsf{P} \subsetneq \Sigma_k$*, pak* $\mathsf{P} \neq \mathsf{NP}$.

2.1 PSPACE-úplnost

Již jsme si ukázali, že PH je celá uvnitř PSPACE. Podobně jako pro NP-úplnost si zavedeme úplnost pro PSPACE:

Definice. Jazyk X je PSPACE-úplný, pokud $X \in$ PSPACE a pro každý $Y \in$ PSPACE je Y je převoditelný na X v polynomiálním čase.

Pro teď budeme předpokládat, že existuje PSPACE-úplný problém L. Později si ukážeme, že takový opravdu existuje.

Pokud by $L \in \Sigma_k$, pak PH spadne do Σ_k.

Důkaz. Nechť $L' \in \Pi_k$. Pak $L' \in$ PSPACE, díky PSPACE-úplnosti je L' převoditelný na L, tedy existuje DTS M běžící v poly čase převádějící L' na $L \in \Sigma_k$. Dále $L \in NP(\Sigma_{k-1})$, tedy existuje NTS akceptor M' s orákulem $A \in \Sigma_{k-1} : L = L(M', A)$.

Zřetězením M a M' dostáváme NTS akceptor M'' pracující v poly čase, pro který $L' = L(M'', A)$. Tudíž $L' \in \Sigma_k$ a $\Pi_k \subseteq \Sigma_k$, dokonce se rovnají a PH kolabuje. □

Důsledek 2.13. *Pokud* PH = PSPACE*, existuje* k*, že* PH $= \Sigma_k$.

Zavedeme si nyní problém, o kterém ukážeme, že je PSPACE-úplný. Jedná se o rozšíření SATu na kvantifikované formule:

Definice (Kvantifikované Bool. formule)**.**

- Pokud je x Booleovská proměnná, pak x je QBF a x je volná.
- Pokud E_1, E_2 jsou QBF, tak $\neg(E_1), (E_1) \wedge (E_2), (E_1) \vee (E_2)$ jsou QBF a status proměnných se nemění.
- Pokud E je QBF, tak $\exists x(E)$ a $\forall x(E)$ jsou QBF. Rozsahem kvantifikátoru jsou volné x v E a tyto výskyty se stávají vázanými.

QBF bez volných proměnných nabývá hodnoty True nebo False. QBF problém dostane na vstupu QBF bez volných proměnných a výstupem je, které hodnoty nabývá.

Vyhodnocování probíhá tak, že pro každé $\forall x(E)$ dosadíme $E_0 \wedge E_1$ a za $\exists x(E)$ dosadíme $E_0 \vee E_1$, kde E_0 je formule E s dosazeným $x = 0$, E_1 s $x = 1$.

Příklad. $\forall x(\forall x(\exists y(x \vee y)) \wedge \neg x)$ je QBF. Všimněme si, že x hlouběji není stejně vázaná proměnná, jako vnější x — můžeme jej přejmenovat na z. Tato QBF je False.

Věta 2.14. *QBF problém (jazyk pravdivých QBF) patří do* PSPACE*.*

Důkaz. Hodnotu vstupního výrazu A počítáme pomocí procedury EVAL:

- Pokud A je konstanta, $\mathrm{EVAL}(A) = A$.
- Pokud $A = A_1 \wedge A_2$, tak $\mathrm{EVAL}(A) = \mathrm{EVAL}(A_1) \wedge \mathrm{EVAL}(A_2)$.
- Pokud $A = A_1 \vee A_2$, tak $\mathrm{EVAL}(A) = \mathrm{EVAL}(A_1) \vee \mathrm{EVAL}(A_2)$.
- Pokud $A = \neg B$, tak $\mathrm{EVAL}(A) = \neg\mathrm{EVAL}(B)$.
- Pokud $A = \forall x(E)$, tak $\mathrm{EVAL}(A) = \mathrm{EVAL}(E_0) \wedge \mathrm{EVAL}(E_1)$.
- Pokud $A = \exists x(E)$, tak $\mathrm{EVAL}(A) = \mathrm{EVAL}(E_0) \vee \mathrm{EVAL}(E_1)$.

Hloubka rekurze EVAL je omezená shora délkou vstupní QBF, dokonce počtem logických operátorů plus kvantifikátorů. Protože vstupní výraz nemá žádné volné proměnné, po rozbalení se jedná o výraz s konstantami. Paměť potřebná na každé vrstvě rekurze je též lineární v délce vstupní QBF. Tudíž QBF je v PSPACE. □

SAT je speciální případ QBF v prenexním tvaru, kde všechny kvantifikátory jsou existenční. Podobně, TAUT má všechny kvantifikátory obecné.

Kvantifikované formule umí zkracovat obecnou formuli (a to až exponenciálně):

Příklad. Formule $f(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} (x_i \Rightarrow y_j)$ se nedá zkrátit a má velikost $\Theta(n^2)$. Ta je splněna právě, když $x_i = 0$ a y_i jsou ohodnoceny libovolně nebo $y_i = 1$ a x_i jsou ohodnoceny libovolně $\forall i$. Takových modelů je $2^{n+1} - 1$.

Zavedeme si novou proměnnou z, kterou „vložíme“ mezi x_i a y_j. Nová formule je tedy $f'(x_1, \ldots, x_n, y_1, \ldots, y_n, z) = \bigwedge_{i=1}^{n} (x_i \Rightarrow z) \wedge \bigwedge_{j=1}^{n} (z \Rightarrow y_j)$. Všechny klauzule $(x_i \Rightarrow y_j)$ jsou logické důsledky, jenže tato funkce má jiné proměnné.

Tudíž nyní uvažme funkci $f''(x_1, \ldots, x_n, y_1, \ldots, y_n)$, definovanou předpisem $\exists z : f'(x_1, \ldots, x_n, y_1, \ldots, y_n, z)$. Ta je již stejná, jako f, ale je kratší – existenční kvantifikátor rozbalíme.

Věta 2.15. *QBF je* PSPACE*-úplný.*

Důkaz. Víme, že QBF je v PSPACE. Nyní ukážeme, že je i úplný.

Nechť $L \in$ PSPACE libovolný jazyk. Ukážeme, že L lze na QBF převést v polynomiálním čase. $L \in \mathrm{DSPACE}(n^c)$ pro pevné c, tedy existuje DTS M pracující v prostoru n^c, kde $L = L(M)$. Tento stroj má nejvýše $c_L^{(n^i)} = 2^{m_L(n)}$ konfigurací, kde $m_L \in \mathcal{O}(n^c)$.

Nyní zavádějme QBF formule:

- $\varphi_{M,x}(C, C')$ jako formuli, která je True právě, když z C do C' jde přejít jedním krokem stroje M. $|\varphi_{M,x}(C, C')|$ je $\mathcal{O}(m_L^2(n))$.
- $\psi_0(C, C') = \varphi_{M,x}(C, C')$
- $\psi_i(C, C')$ je True právě, když z C do C' existuje výpočet délky 2^i kroků (v grafu konfigurací existuje cesta délky 2^i z C do C').
- $\psi_{m_L(n)}(C_{\mathrm{START}}, C_{\mathrm{END}}) = 1 \Leftrightarrow x \in L$.

Jak vypadá $\psi_i(C, C')$? Definice $\exists C'' : (\psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C'))$ nefunguje, protože formule roste exponenciálně. Musíme na to jinak:

$\exists C'' \forall D, D' : (((D = C) \wedge (D' = C'')) \vee ((D = C'') \wedge (D' = C')) \Rightarrow \psi_{i-1}(D, D'))$.

Nyní již platí $|\psi_{m_L(x)}| \in \mathcal{O}(m_L^2(n))$. A protože tato formule je pravdivá právě, když $x \in L$, jsme hotovi. □

Proč nám nestačí omezený počet alternací kvantifikátorů? Jejich počet závisí lineárně na hloubce rekurze. Jenže tato hloubka závisí na vstupu. Tudíž, QBF nepatří do Δ_c pro libovolné $c > 0$.

2.2 Úplnost v PH

Ukážeme si, že pro každou úroveň polynomiální hiearchie existují úplné problémy. Bude se jednat o kvantifikované formule s omezeným počtem kvantifikátorů.

Definice. Nechť φ je CNF a každé u_i je vektor Booleovských proměnných. Pak Σ_i-SAT : $\exists u_1 \forall u_2 \dots Q u_i : \varphi(u_1, \dots, u_i) = 1$ a Π_i-SAT : $\forall u_1 \exists u_2 \dots Q u_i : \varphi(u_1, \dots, u_i) = 1$ kde Q je odpovídající kvantifikátor podle střídání.

Věta 2.16. *Σ_i-SAT je Σ_i-úplný a Π_i-SAT je Π_i-úplný.*

Důkaz. Σ_i-SAT $\in \Sigma_i$ a Π_i-SAT $\in \Pi_i$ plyne přímo z definice. Dokážeme, že Σ_i-SAT je úplný, pro Π_i-SAT je důkaz obdobný.

Nechť $L \in \Sigma_i$ je libovolný jazyk. Pak existuje $L' \in \mathsf{P}$ a polynom p takový, že $x \in L \Leftrightarrow \exists^{p(|x|)} y_1 \forall^{p(|x|)} y_2 \dots Q^{p(|x|)} y_i : (x, y_1, \dots, y_i) \in L'$.

Protože $L' \in \mathsf{P}$, existuje DTS M takový, že rozpoznává L' v polynomiálním čase. Stejně, jako v důkazu Cook-Levinovy věty lze výpočet stroje M nad vstupem x zakódovat do CNF formule φ_M tak, že $\varphi_M(w) = 1$ právě, když $w \in L'$ pro každé $w = (x, y_1, \dots, y_i)$.

Nyní $x \in L \Leftrightarrow \exists^{p(|x|)} y_1 \forall^{p(|x|)} y_2 \dots Q^{p(|x|)} y_i : (x, y_1, \dots, y_i) \in L' \Leftrightarrow \exists^{p(|x|)} y_1 \forall^{p(|x|)} y_2 \dots Q^{p(|x|)} y_i : \varphi(x, y_1, \dots, y_i) = 1$. □

Jeden by mohl namítnout, že $x, y_1, \dots, y_n$ jsou v jiné abecedě. Ale nic nám nebrání je překódovat do správné abecedy. Může se nám změnit délky, ale nejvýše konstanta krát.

3 Převoditelnost v logaritmickém prostoru

Které jazyky jsou P-úplné vzhledem k převoditelnosti v polynomiálním čase? Pokud $A \neq \emptyset$ nebo $A \neq \Sigma^*$ a navíc $A \in \mathsf{P}$, je A P-úplný. Taková definice P-úplnosti je nám trochu nanic.

Definice. Jazyk A je P-úplný, pokud $A \in \mathsf{P}$ a $\forall B \in \mathsf{P}$ $\exists$ DTS transducer M_B pracující v logaritmickém prostoru takový, že $x \in B \Leftrightarrow M_B(x) \in A$.

Věta 3.1. *Nechť A je P-úplný. Pak pokud $A \in \mathsf{L}$, pak $\mathsf{P} = \mathsf{L}$.*

Důkaz. Dle věty o vztazích již víme, že $\mathsf{L} \subseteq \mathsf{P}$.

Nechť $B \in \mathsf{P}$ je libovolný. Protože A je P-úplný, existuje DTS M pracující v logaritmickém prostoru převádějící B na A. Navíc $A \in \mathsf{L}$, tudíž existuje DTS M' pracující v logaritmickém prostoru přijímající A.

Jednoduché zřetězení M a M' nefunguje, protože zřetězením pracovní prostor obsahuje pracovní pásku M, řetězec y jakožto vstup M' a pracovní páska M'. Dále víme, že $|y| \leq |x|^{\log c}$; máme jen tolik času. Ten se nám nevejde do logaritmického prostoru.

Stroj M'' proto vznikne zřetězením M a M' s tím, že na svých páskách obsahuje pracovní prostory M a M' a y je generováno znak po znaku vždy, kdy je potřeba nový symbol pod vstupní hlavou M'. K tomu bude potřebovat dvě počítadla délky $\log y \in \mathcal{O}(\log x)$.

První počítadlo obsahuje aktuální pozici vstupní hlavy M'. Pokud M' pohne vstupní hlavou, první počítadlo je příslušně aktualizováno a M je puštěn od začátku s tím, že pozice aktuálně generovaného znaku y je držena ve druhém počítadle. Znaky zahazuje, dokud se obě počítadla neshodnou.

Stroj M'' pracuje v logaritmickém prostoru a přijímá B, tudíž $B \in \mathsf{L}$. □

Jak si pamatujeme z PH, pokud máme NP-úplný problém L a navíc $L \in$ Co-NP, pak NP = Co-NP. Navíc, pokud toto nastane, celá PH zkolabuje do NP. Něco podobného máme pro NL. Avšak narozdíl od PH najdeme takový problém, speciálně PATH.

Lemma 3.2. *Nechť $L \in$ NLÚ. Pokud navíc $L \in$ Co-NL, L = NL.*

Věta 3.3 (Szelepcsenyi-Immerman)**.** NL = Co-NL.

Důkaz. Jazyk PATH $= \{(G, s, t) \mid$ G je orientovaný graf, ve kterém $\exists$ cesta z s do $t\}$. Chceme ukázat:

1. PATH $\in$ NL: Nedeterministicky hádáme sled, pamatujeme si jeho délku a poslední vrchol. Nejprve zapíšeme délku 0 a vrchol s. V následujícím kroku přejdeme na souseda v nedeterministicky a přičteme délku sledu o 1. Pokud narazíme na t, končíme kladně. Pokud délka sledu překročí počet vrcholů G, končíme záporně.

   Protože existence sledu implikuje existenci cesty, stačí procházet sledy délky nejvýše $|V(G)|$.

2. PATH je NL-úplný: Nechť $A \in$ NL libovolný. Chceme ukázat, že existuje DTS transducer M' pracující v logaritmickém prostoru převádějící A na PATH. Víme, že existuje NTS M rozpoznávající A v logaritmickém prostoru. BÚNO M má právě jednu přijímající konfiguraci.

   Stroj M' vygeneruje pro vstup x na svou výstupní pásku popis grafu G všech konfigurací M plus počáteční a přijímající konfiguraci. To je zřejmě hledaný převod, musíme ale ukázat, že M' pracuje v logaritmickém prostoru.

Popis G lze generovat takto: Systematicky (lexikograficky) M' generuje jednotlivé konfigurace M. Ke každé vygenerované konfiguraci k stroj M' přepíše z popisu M, který je konstantně velký, všechny sousedy k. Je nutné si pamatovat poslední konfiguraci, která se jistě vejde do $\mathcal{O}(\log |x|)$.

3. PATH $\in$ Co-NL: Stačí umět nedeterministicky spočítat počet dosažitelných vrcholů z s; spočítáme počty dosažitelné vrcholů α pro (G, s, t) a β pro (G', s, t), kde G' vznikl z G odstraněním t; kdykoliv narazíme na souseda t, větev ignorujeme. Nakonec přijmeme, pokud $\alpha = \beta$ (počet dosažitelných vrcholů se nezměnil, t nemůže být dosažitelný).

   Jak tedy spočítat počet dosažitelných vrcholů? Nechť R_i je množina vrcholů dosažitelných ze s cestou délky nejvýše i. Pak $R_0 = \{s\}$ a $R_i = R_{i-1} \cup \{v \mid \exists u \in R_{i-1} : (u, v) \in E(G)\}$. Budeme nedeterministicky počítat $|R_0|, |R_1|, \ldots$ s tím, že si budeme pamatovat jen poslední číslo – potřebujeme spočítat $|R_i|$ ze znalosti $|R_{i-1}|$.

   Nedeterministicky hádáme odhad g pro $|R_i|$ (zkusíme čísla mezi $|R_{i-1}|$ a n) a následně ověříme, že $|R_i| \geq g$ a $|R_i| \leq g$.

   $|R_i| \geq g$: nedeterministicky vybereme právě g vrcholů a pro každý z nich nedeterministicky ověříme existenci cesty z s délky nejvýše i. Pamatujeme si poslední vrchol, čítač vybraných vrcholů a prostor pro ověření nalezení cesty, vše logaritmické. Vrcholy systematicky procházíme a rozhodneme, zda jej vybereme – přičteme do čítače a ověříme. Nakonec je potřeba ověřit, že jich bylo vybráno g.

   $|R_i| \leq g$ je ekvivalentní tomu, že $|V \setminus R_i| \geq n - g$. Generujeme systematicky $n - g$ vrcholů. Pro každý vybraný vrchol w projdeme ostatní vrcholy a nedeterministicky jich vybereme právě $|R_{i-1}|$. Pro každý vybraný vrchol v ověříme $(v, w) \notin E$ a $v \in R_{i-1}$. Vše zvládneme v logaritmickém prostoru – potřebujeme kódy a čítače w, v a logaritmický prostor na ověření $v \in R_i$.

Teď už stačí jen použít předchozí lemma. □

4 Neuniformní výpočetní modely

Až doteď jsme si povídali o uniformních modelech, které se dokázaly vypořádat se všemi vstupy. Nyní se ale zaměříme na modely, které umí pracovat jen s omezeným množstvím vstupů – například vstupy konkrétní délky.

Definice. *Booleovský obvod* s n vstupy a jedním výstupem je acyklický graf s n *zdroji* a jedním *stokem*.

Zdrojové vrcholy jsou vstupy (binární), ostatní vrcholy jsou hradla (AND, OR, NOT). jenž mají aritu 2 a v případě NOT aritu 1.

Velikostí Booleovského obvodu se rozumí počet vrcholů.

Příklad. Formuli XOR, v DNF reprezentovanou $(\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$ odpovídá obvod, jehož velikost je 7 (výstupy sdílí jako vstup více hradel).

Příklad. Formule v CNF $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee x_5)$ lze převést přímočaře, tak dostaneme obvod velikosti 13. Takový obvod má u hradel výstupní stupně nejvýše 1.

Avšak můžeme postupovat lépe – část formule zrecyklujeme, jelikož nic nám neomezuje výstupní stupeň. Konkrétně zde můžeme použít výstup $x_1 \vee x_2$ třikrát jen jako jedno hradlo, čímž snížíme velikost obvodu na 11.

Rozmysleme si způsoby, jak relaxovat model a jak taková relaxace ovlivní velikost obvodu:

- Zrušíme omezení arity AND, OR. Pak každé hradlo v obvodu C s aritou k nahradíme $k-1$ binárními hradly (kaskáda nebo strom). Tím vznikne obvod C' s hradly arity nejvýše 2 a $|C'| \leq |C|^2$.
- Pokud je výstupní arita vrcholů mimo stok rovna jedné, pak obvody přímo odpovídají formulím. Jejím zvednutím můžeme dostat obvody až logaritmicky menší.

  Typickým příkladem je problém parity. Zatímco formule je exponenciálně dlouhá (popisujeme v DNF všechny množiny liché velikosti), obvod je lineární $(x_1 \oplus x_2 \oplus \cdots \oplus x_n)$.

Definice. Vyhodnocením vstupu $x \in \{0,1\}^n$ obvodem C (s n vstupy) rozumíme hodnotu $C(x)$ výstupního vrcholu, pokud do vstupních vrcholů dosadíme x.

Hodnota výstupu je definovaná přirozeným rekurzivním způsobem.

Každou abecedu můžeme převést na binární. Tím můžeme použít booleovské obvody jako výpočetní model.

Definice. Nechť $T : \mathbb{N} \to \mathbb{N}$ je funkce. Rodina obvodů velikosti $T(n)$ je posloupnost obvodů $\{C_n\}_{n\in\mathbb{N}}$, kde $\forall n : C_n$ má n vstupů a platí $|C_n| \leq T(n)$.

Jazyk L patří do třídy SIZE($T(n)$), pokud existuje rodina obvodů $\{C_n\}_{n\in N}$ velikosti $T(n)$ taková, že $\forall n \forall x \in \{0,1\}^n : x \in L \Leftrightarrow C_n(x) = 1$.

Příklad. Jazyk $L = \{1^n \mid n \in \mathbb{N}\} \in \text{SIZE}(\mathcal{O}(n))$ – stačí kaskáda AND hradel, tudíž $|C_n| = 2n - 1$.

$L = \{(a, b, a + b) \mid a, b \in \mathbb{N}\} \in \mathrm{SIZE}(\mathcal{O}(n))$, $n = 3\lceil \log(\max(a, b) + 1) \rceil$. Obvod sestavíme jako binární sčítačku na prvních $2n/3$ bitech nakonec porovnáme s posledními $n/3$ bity.

Definice. P/poly je třída jazyků přijímaných rodinami obvodů polynomiální velikosti, tedy $\mathsf{P/poly} = \bigcup_{c=0}^{\infty} \mathrm{SIZE}(n^c)$.

Třída $\mathsf{P/poly} \subsetneq \mathsf{P}$. Představme si jazyk $L \subseteq 1^n$, kde navíc n je kód dvojice (M, x), kde M je DTS a x je vstup a $M(x)\downarrow$. Pro takový jazyk jistě existují obvody (kaskáda ANDů pro n správné, nepravdu pro ostatní). Tyto obvody ale neumíme efektivně vygenerovat.

Definice. DTS je *oblivious*, pokud má jednu vstupní a pracovní pásku a pozice hlav nezávisí na vstupu, ale pouze na počtu kroků od začátku výpočtů a délce vstupu.

Pokud existuje normální DTS M, pak rovněž existuje oblivous DTS M'. Stroj M budeme na M' simulovat tak, že každý krok budeme simulovat přejezdem hlavy přes celý možný prostor, přičemž provedeme lokální změny. Tím můžeme prodloužit čas výpočtu M až kvadraticky.

Věta 4.1. $\mathsf{P} \subseteq \mathsf{P/poly}$.

Důkaz. Mějme jazyk $L \in \mathsf{P}$. Tudíž existuje DTS M pracující v polynomiálním čase takový, že $L = L(M)$. Pak též existuje oblivous DTS M' v poly čase, kde $L = L(M')$.

Ukážeme, jak pro pevné L zkonstruovat obvod C_n přijímající všechna slova z L délky n. Displej M' je trojice (stav, vstupní symbol, pracovní symbol), můžeme jej tedy překódovat do řetězce.

Existuje konstantně velký obvod C_i, který na základě displeje po $(i-1)$-tém kroku M' vydá displej po i-tém kroku. Protože hlava se mohla posunout, musíme nějak poslat informaci o tom, který znak byl přečten.

Navíc bude C_i kromě displeje potřebovat vstup x a $C_{i_v}, C_{i,p}$, kde i_v, i_p je číslo kroku, kdy byl stroj M' naposledy na příslušné pozici vstupu nebo pracovní pásky. Protože M' je oblivious, i_v, i_p jsou závislé pouze na i.

Předpokládejme, že M' běží v čase $T(n)$. Pak obvod $C_{T(n)}$ vydá True právě, když displej po kroku $T(n)$ obsahuje kód přijímajícího stavu. Tento obvod přijímá právě slova z L délky n a je polynomiálně velký. □

Důsledek 4.2. *Příslušná rodina $\{C_n\}$ nejen existuje, ale lze ji efektivně generovat: existuje DTS $\hat{M}$, který na vstup 1^n vydá popis obvodu C_n a navíc pracuje v polynomiálním čase a logaritmickém prostoru.*

Proč logaritmický prostor? $\hat{M}$ musí být schopen z i spočítat i_v a i_p, takže potřebuje $\mathcal{O}(\log n)$ prostoru. Samotné obvody jsou konstantně velké, takže je lze generovat inkrementálně v konstantním prostoru.

Definice. Rodina obvodů $\{C_n\}$ je P-uniformní, pokud existuje DTS, který na vstupu 1^n vydá v polynomiálním čase na výstupu popis C_n.

Věta 4.3. *L je přijímán P-uniformní rodinou obvodů právě, když $L \in \mathsf{P}$.*

Důkaz. Nechť L je přijímán $\{C_n\}$, které je P-uniformní. Pak L je přijímán DTS M, které nejprve simuluje chod DTS generujícího C_n na vstupu 1^n, kde $n = |x|$. Následně M simuluje práci C_n na vstupu x.

Druhá implikace vychází z posledního důsledku — $\mathsf{P} \subseteq \mathsf{P/poly}$ — sestavíme $\{C_n\}$ rozpoznávající L, který je P-uniformní. □

Definice. Rodina obvodů $\{C_n\}$ je log-space uniformní, pokud existuje DTS, který na vstupu 1^n vydá v logaritmickém prostoru na výstupu popis C_n.

Věta 4.4. *L přijímá log-space uniformní rodina obvodů právě, když $L \in \mathsf{P}$.*

Důkaz. Stejný, jako v předchozí větě. Stroj, který pracuje v logaritmickém prostoru, pracuje i v polynomiálním čase. □

Definice. Jazyk CIRCUIT-EVAL je jazyk dvojic (C, x), kde C je popis n-vstupového booleovského obvodu a $x \in \{0,1\}^n$, takových, že $C(x) = 1$.

Věta 4.5. *Jazyk* CIRCUIT-EVAL *je P-úplný.*

Důkaz. Nechť $L \in \mathsf{P}$. Pak log space transducer, který převádí jazyk L na jazyk CIRCUIT-EVAL, je implicitně schován v druhé implikaci předchozí věty. □

Definice. Řetězec reprezentující obvod C na n vstupech patří do jazyka CCT-SAT právě, když $\exists u \in \{0,1\}^n : C(n) = 1$.

Lemma 4.6. CCT-SAT $\in \mathsf{NP}$.

Důkaz. Certifikát pro $C \in$ CCT-SAT je $u \in \{0,1\}^n$ takové, že $C(n) = 1$. □

Lemma 4.7. CCT-SAT *je NP-těžký.*

Důkaz. Nechť $L \in \mathsf{NP}$ je libovolný jazyk. Pak existuje DTS M pracující v polynomiálním čase takový, že $x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} : p(x,u) = 1$. V důkazu věty $\mathsf{P} \subseteq \mathsf{P/poly}$ plyne, že ke stroji M lze zkonstruovat polynomiálně velkou rodinu přijímající L.

Ke vstupu (x,u) délky $m = n + p(n)$ existuje obvod C_m, pro který platí: $M(x,u) = 1 \Leftrightarrow C_m(x,u) = 1$. Pro pevný vstup x vyrobíme z C_m obvod C_m^x

zafixováním vstupů x na konstanty. Nyní $x \in L \Leftrightarrow \exists u : M(x,u) = 1 \Leftrightarrow \exists u : C_m(x,u) = 1 \Leftrightarrow \exists u : C_m^x(x,u) = 1 \Leftrightarrow C_m^x \in$ CCT-SAT. □

Věta 4.8. CCT-SAT *je polynomiálně převoditelný na* 3-SAT.

Důkaz. Nechť C je obvod (instance CCT-SAT) se vstupy $x_1, \ldots, x_n$ a hradly $y_1, \ldots, y_n$ (každé y_i je buď AND, OR nebo NOT). Sestavíme 3CNF F_C takovou, že je splnitelná právě, když $C \in$ CCT-SAT:

- Nechť y_i je hradlo NOT se vstupem z_j. Pak přidáme klauzule $(z_j \vee y_i) \wedge (\neg z_j \vee \neg y_i)$.
- Nechť y_i je hradlo AND se vstupy z_j, z_k. Pak přidáme klauzule $(\neg y_i \vee z_j) \wedge (\neg y_i \vee z_k) \wedge (y_i \vee \neg z_j \vee \neg z_k)$.
- Nechť y_i je hradlo OR se vstupy z_j, z_k. Pak přidáme klauzule $(\neg y_i \vee z_j \vee z_k) \wedge (y_i \vee \neg z_j) \wedge (y_i \vee \neg z_k)$.
- Nechť y_m je výstupní hradlo. Pak přidáme unární klauzuli (y_m).

Spojení těchto klauzulí vydá korektní 3CNF F_C, která je splnitelná právě, když původní obvod C je splnitelný. □

Formule v důkazu se taktéž nazývá Tseitinovo kódování obvodu.

4.1 Turingovy stroje s radící funkcí

Nechť M je DTS a $\{\alpha_n\}_{n\in\mathbb{N}}$ je posloupnost řetězců. Pokud M pracuje nad vstupem x délky n, tak má přístup k „radě“ α_n.

Definice. Nechť $T, a : \mathbb{N} \to \mathbb{N}$ jsou funkce. Pak třída $\mathrm{DTIME}(T(n))|_{a(n)}$ je třída jazyků rozpoznatelná DTS v čase $T(n)$ s $a(n)$-bitovou radou.

Formálně $L \in \mathrm{DTIME}(T(n))|_{a(n)}$ právě, když existuje DTS M a posloupnost $\{\alpha_n\}$, kde $\alpha_n \in \{0,1\}^{a(n)}$, pro která platí $x \in L \leftrightarrow M(x, \alpha_n) = 1$ pro všechny $x \in \{0,1\}^n$ s tím, že M na vstupu (x, α_n) udělá $\mathcal{O}(T(n))$ kroků.

Věta 4.9. $\mathsf{P/poly} = \bigcup_{c,d} \mathrm{DTIME}(n^c)|_{n^d}$.

Důkaz. Nechť $L \in \mathsf{P/poly}$. Z předchozího víme, že L je rozpoznávaný polynomiálně velkou rodinou obvodů $\{C_n\}$ a L je rozpoznáván DTS M, který na vstupu x délky n vezme popsi C_n jako radu a odsimuluje chod C_n na x.

Nechť $L \in \bigcup_{c,d} \mathrm{DTIME}(n^c)|_{n^d}$. Pak je L rozpoznáván DTS M pracující v polynomiálním čase $(p(n) = n^c)$, který má přístup k radám $\{\alpha_n\}$ velikosti $|\alpha_n| = n^d$. Zopakováním důkazu $\mathsf{P} \subseteq \mathsf{P/poly}$ zkonstruujeme k DTS

M polynomiálně velkou rodinu obvodů $\{D_n\}$ takovou, že pro každou dvojici $(x, \alpha) : x \in \{0,1\}^n, \alpha \in \{0,1\}^{a(n)} \Rightarrow D_n(x, \alpha) = M(x, \alpha)$.

Nyní z rodiny obvodů $\{D_n\}$ vyrobíme rodinu $\{C_n\}$ tak, že pro každé n zafixujeme vstup α. Pak $\forall x \in \{0,1\}^n : C_n(x) = M(x, \alpha)$. □

Věta 4.10. $\forall n > 1\, \exists f : \{0,1\}^n \to \{0,1\}$ *taková, že* f *nemůže být počítána obvodem velikosti nejvýše* $2^n/10n$.

Důkaz. Všech booleovských funkcí na n proměnných je 2^{2^n}.

Mějme booleovský obvod s m vrcholy. Při omezeném vstupním stupni nejvýše 2 máme nejvýše $2m$ hran. Popis takového obvodu se vejde do $3m \log m$ bitů.

Počet různých obvodů s m vrcholy je nejvýše $2^{3m \log m}$. Po dosazení $m = 2^n/10n$ dostáváme $2^{3 \cdot 2^n/10m \log 2^n/10n} \leq 2^{3 \cdot 2^n/10n \cdot n} = 2^{3/10 \cdot 2^n} < 2^{2^n}$. □

Byla snaha (která se nepovedla) najít funkci v NP takovou, že příslušná nejde spočítat polynomiálně velkými obvody. Tím by se mohlo dokázat $\mathsf{P} \subsetneq \mathsf{NP}$.

Naopak, pokud $\mathsf{NP} \subseteq \mathsf{P/poly}$, potom můžeme ukázat, že PH zkolabuje.

Věta 4.11 (Lipton-Karp)**.** *Pokud* $\mathsf{NP} \subseteq \mathsf{P/poly}$*, pak* $\mathsf{PH} = \Sigma_2 = \Pi_2$.

Důkaz. Ukážeme, že Π_2-SAT $\in \Sigma_2$. Podle definice Π_2-SAT je jazyk formulí $\varphi : \forall u \in \{0,1\}^n\, \exists v \in \{0,1\}^m : \varphi(u, v) = 1$. Při zafixování u dostáváme problém ve $\Sigma_1 = \mathsf{NP} : \exists v \in \{0,1\}^m : \varphi(u, v) = 1$. Jinými slovy máme jazyk $L = \{(\varphi, u) \mid \exists v : \varphi(u, v) = 1\}$, jehož certifikát je v.

Protože $\mathsf{NP} \subseteq \mathsf{P/poly}$, existuje polynomiálně velká rodina obvodů $\{C_k\}$ velikosti $p(k)$ taková, že pro každou CNF φ v $n + m$ proměnných a každé $u \in \{0,1\}^n$ platí $C_k(\varphi, u) \Leftrightarrow \exists v \in \{0,1\}^n : \varphi(u, v) = 1$. Tudíž k je n plus počet bitů na zakódování φ.

Nyní převedeme $\{C_k\}$ na rodinu $\{C'_k\}$ obvodů s $m + 1$ výstupy, kde první výstup je stejný, jako u C_k, a pokud je tento výstup 1, tak zbylých m výstupů obsahuje nějaké v takové, že $\varphi(u, v) = 1$. Na to půjdeme tak, že budeme postupně zjišťovat jednotlivé ohodnocení proměnných na základě odpovědi φ s částečným ohodnocením v, které rozšiřujeme.

Jistě platí $|C'_k| \leq q(k)$ pro vhodný polynom q. Jistě lze C'_k zakódovat do řetězce délky $\mathcal{O}(q^2(k)) = r(k)$ a tento řetězec lze „hádat“ pomocí formule $\psi : \exists w \in \{0,1\}^{r(k)}\, \forall u \in \{0,1\}^n : \varphi(u, C'_k(u, v)) = 1$, kterou navíc pro dané w, u spočítáme v polynomiálním čase, jazyk pravdivých formulí ψ patří do Σ_2-SAT.

- Nechť φ je takové, že $\forall u \in \{0,1\}^n\, \exists v \in \{0,1\}^m : \varphi(u, v) = 1$. Pak obvod $C'_k(\varphi, u)$ nějaké takové v vrátí, tudíž platí i ψ.

- Nechť φ je takové, že $\exists u \in \{0,1\}^n \, \forall v \in \{0,1\}^m : \varphi(u,v) = 0$. Pak není šance, aby platilo ψ.

Tudíž máme korektní převod a Π_2-SAT $\in \Sigma_2$. □

4.2 Třídy NC a AC

Definice. Jazyk $L \in \mathsf{NC}^d$, pokud existuje polynomiálně velká rodina obvodů $\{C_n\}$ rozpoznávající L taková, že $\forall n$ hloubka C_n je $\mathcal{O}(\log^d n)$.

Definice. Třída $\mathsf{NC} = \bigcup_{d \geq 0} \mathsf{NC}^d$.

Definice. Stejně definujeme třídy AC^d a AC s tím rozdílem, že hradla zde mají povolen libovolný počet vstupů.

Lemma 4.12. $\forall d : \mathsf{NC}^d \subseteq \mathsf{AC}^d \subseteq \mathsf{NC}^{d+1}$.

Důkaz. První inkluze je triviální. Nechť $L \in AC^d$. Každé hradlo má maximálně $p(n)$ vstupů. Tím pádem, když toto hradlo nahradíme binárním stromem binárních hradel, má tento strom hloubku $\log p(n) \in \mathcal{O}(\log n)$. Celková hloubka se tak zvýší nejvýše logaritmicky. □

Zaveďme si nový model – paralelní počítač. Ten bude mít n procesorů, komunikační kanály mezi nimi, vše taktováno stejnými hodinami a vzdálenost mezi procesory je $\mathcal{O}(\log n)$ (například hyperkrychlová architektura).

Definice. Výpočetní úloha má efektivní paralelní algoritmus, pokud zadání velikosti n lze vyřešit na paralelním počítači s $n^{\mathcal{O}(1)}$ procesory v čase $\log^{\mathcal{O}(1)} n$.

Jazyk L má efektivní paralelní algoritmus právě, když $L \in \mathsf{NC}$:

L je přijímán paralelním počítačem, který pro vstup délky n má $N \in \mathcal{O}(n^c)$ procesorů a pracuje v čase $D \in \mathcal{O}(\log^d n)$. Zkonstruujeme obvod C_n sestávající z ND obvodů konstantní velikosti uspořádaných do D hladin o N hradlech. Každý obvod simuluje práci konkrétního procesorů v konkrétním taktu daného výpočtu. Tudíž $L \in \mathsf{AC}^d \subseteq \mathsf{NC}$.

Naopak máme obvod s n uzly, který nahradíme počítačem s n procesory. Hloubka obvodu $\log^d n$ může být simulovaná v čase $\log^{d+1} n$, protože lze spojit dvě libovolné hradla, ale v paralelním počítači může signál putovat až $\log n$ čas.

5 Randomizované výpočty

Do teď jsme uvažovali pouze deterministické a naopak plně nedeterministické stroje. NTS neumíme naimplementovat. Nabízí se tedy omezit nedeterminismus tak, aby stále přidával sílu oproti deterministické verzi, ale byl prakticky použitelný.

Definice. Pravděpodobnostní Turingův stroj je TS se dvěma přechodovými funkcemi δ_1, δ_2. Při práci PTS M na vstupu x v každém kroku s pravděpodobností $1/2$ použije δ_1 nebo δ_2. Každý výběr je nezávislý na všech předchozích. M vrací hodnotu 1 (x přijato) nebo 0 (x zamítnuto) a tuto náhodnou veličinu značíme $M(x)$.

Říkáme, že M pracuje v čase $T(n)$, pokud pro každý vstup x velikosti n odpoví stroj M po nejvýše $T(n)$ krocích bez ohledu na výběru δ_1, δ_2.

Zatímco NTS přijímá, pokud nějaká větev výpočtu přijme s neznámou pravděpodobností, u PTS je pravděpodobnost přijetí přímo poměr přijímajících větví vůči všem.

Pro jazyk $L \subseteq \{0,1\}^*$ a $x \in \{0,1\}^*$ definujeme $L(x) = 1$ právě, když $x \in L$, a naopak $L(x) = 0$ právě, když $x \notin L$.

Definice. Řekněme, že PTS M přijímá jazyk $L \subseteq \{0,1\}^*$ v čase $T(n)$, pokud $\forall x \in \{0,1\}^* : \Pr[M(x) = L(x)] \geq 2/3$ a M běží v čase $T(n)$.

BPTIME($T(n)$) je třída jazyků přijímaných PTS v čase $\mathcal{O}(T(n))$.

$\mathsf{BPP} = \bigcup_{c \geq 0} \mathrm{BPTIME}(n^c)$ (Bounded-error Probabilistic Polynomial Time).

Třída $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{EXP}$, stačí zvolit $\delta_1 = \delta_2$ a projít všechny větve PTS.

Podobně jako NP můžeme též BPP definovat jako třídu, kterou přijímají DTS, ke kterým se na vstup přidají náhodné bity.

Konstanty v definicích nemusí být konkrétní. Pro BPP funguje libovolná konstanta z intervalu $(1/2, 1)$. Speciálně pro BPP ukážeme, jak redukovat chybu nad libovolnou konstantu.

Věta 5.1. *Nechť $L \subseteq \{0,1\}^*$ je jazyk, $c > 0$ konstanta a předpokládejme, že existuje PTS M pracující v polynomiálním čase takový, že $\forall x \in \{0,1\}^* : \Pr[M(x) = L(x)] \geq 1/2 + |x|^{-c}$.*

Potom $\forall d > 0$ existuje PTS M' pracující v polynomiálním čase takový, že $\forall x \in \{0,1\}^* : \Pr[M'(x) = L(x)] > 1 - 2^{|x|^d}$.

K důkazu budeme potřebovat Chernoffovy nerovnosti: $X_1, \ldots, X_n \in \{0,1\}$ nezávislé náhodné proměnné, $X = \sum X_i$ a $\mu = E[X]$.

Pak $\forall c > 0 : \Pr[|\sum_i X_i - \mu| \geq c \cdot \mu] < 2e^{-\mu \min(c^2/4, c/2)}$.

Důkaz. M' pracuje takto na vstupu x: Pustí M opakovaně k-krát pro $k = 8|x|^{2c+d}$, čímž získá k binárních výsledků $y_1, \ldots, y_k$. Pokud je $\sum y_i \geq k/2$, pak M' přijme x, v opačném případě odmítne x. Pokud M pracuje v čase $t(x)$, pak M' pracuje v čase $t'(x)$, kde t, t' jsou polynomy.

Označme náhodné proměnné $X_i \in \{0, 1\}$ takové, že $X_i = 1 \Leftrightarrow y_i = L(x)$ a $X = \sum_i X_i$. Zjevně X_i jsou nezávislé náhodné proměnné, pro které $E[X_i] = \Pr[X_i = 1] \geq p = 1/2 + |x|^{-c}$. Zvolme $\delta = 1/2 \cdot |x|^{-c}$ a uvažujme nerovnost $\sum_i X_i \geq pk - \delta pk = kp(1 - \delta) = k/2 + k(3/4 \cdot |x|^{-c} - 1/2 \cdot |x|^{-2c}) > k/2$.

Platnost této nerovnosti garantuje, že M' odpověděl zaručeně správně. Tudíž M' může udělat chybu, pokud platí $\sum_i X_i < pk - \delta pk$. To též znamená $pk - \sum X_i > \delta pk$, což nastane s pravděpodobností $\Pr[pk - \sum X_i > \delta pk]$. Víme, že $E[X_i] \geq p$, pak $\mu \geq pk$. Označme $\mu = pk + \varepsilon$ pro $\varepsilon > 0$.

Pak $\Pr[pk - \sum X_i > \delta pk] \leq \Pr[pk - \sum X_i + (1 - \delta)\varepsilon > \delta pk] = \Pr[pk + \varepsilon - \sum X_i > \delta pk + \delta\varepsilon] \leq \Pr[|\sum X_i - \mu| > \delta\mu] \leq 2e^{-\mu\delta^2/4}$. Dosadíme zpět hodnoty a dostáváme $2e^{-1/4 \cdot |x|^d - 1/2 \cdot |x|^{d-c}} \leq 2e^{-1/4 \cdot |x|^d}$, což je asymptoticky, co chceme. To je navíc horní odhad chyby. □

Definice BPP je robustní vůči generování náhodných bitů pomocí jakékoliv mince, ne nutně rovnoměrné.

Lemma 5.2. *Mince s* $\Pr[\text{hlava}] = \rho, \rho \in (0, 1)$ *lze simulovat na PTS* M *v očekávaném čase* $\mathcal{O}(1)$*, pokud* i*-tý bit* ρ *lze generovat v čase polynomiálním vzhledem k* i*.*

Důkaz. Nechť $\rho = 0.p_1p_2\ldots$ je binární zápis ρ. PTS M generuje rovnoměrné nezávislé náhodné bity $b_1, b_2, \ldots$ (v čase i generuje b_i). Pokud v i-tém kroku:

- $b_i < p_i$: M vrací „hlavu“ a zastaví.
- $b_i > p_i$: M vrací „orel“ a zastaví.
- $b_i = p_i$: M pokračuje krokem $i + 1$.

Nyní pravděpodobnost, že M vrátí „hlavu“ v kroku i je $2^{-i+1} \cdot k$, kde první člen je rovnost bitů v prvních $i - 1$ krocích a nerovnost v i-tém. Pokud $b_i = 1$, pak $k = 1/2$, jinak $k = 0$. Tudíž pravděpodobnost, že M vrátí „hlavu“ je $\sum_{i=1}^{\infty} p_i \cdot 2^{-i} = \rho$.

Očekávaný čas výpočtu je nejvýše $\sum_{i=1}^{\infty} 2^{1-i} \cdot i^c$. To ale shora můžeme odhadnout konstantou. □

Lemma 5.3. *Rovnoměrnou minci lze naopak simulovat pomocí PTS s přístupem k sekvenci bitů pomocí mince s* $\Pr[\text{hlava}] = \rho$ *v očekávaném čase* $\mathcal{O}(1)$*.*

Důkaz. PTS M dvakrát hodí mincí po sobě. Pokud se výsledky těchto dvou mincí liší, jako výstup prohlásíme hodnotu prvního hodu. Jinak v případě rovnosti opakujeme. Pak $\Pr[\text{hlava}] = \Pr[\text{orel}] = \rho(1-\rho)$.

V každém kroku se M zastaví s pravděpodobností $2\rho(1-\rho)$. Tudíž střední počet kroků bude $\sum_{i=1}^{\infty} i \cdot c^i \in \mathcal{O}(1)$. □

Definice. Nechť M je PTS a x je vstup. Pak $T_{M,x}$ je délka běhu M na vstupu x a je to náhodná veličina závisející na náhodných bitech určujících výpočet.

Třída BPP je taktéž robustní vzhledem k „měření času“: místo požadavku, že PTS skončí po nejvýše $p(|x|)$ mnoha krocích, stačí požadovat, že očekávaná doba výpočtu je nejvýše $p(|x|)$.

Definice. M běží v očekávaném čase $T(|x|)$, pokud $\forall x \in \{0,1\}^* : E[T_{M,x}] \leq T(|x|)$.

Propůjčme si další nerovnost z teorie pravděpodobnosti, a to Markovovu. Nechť X je náhodná veličina a μ její střední hodnota. Pak $\Pr[X \geq k\mu] \leq 1/k$.

Lemma 5.4. *Nechť PTS M běží v očekávaném čase $p(|x|)$. Pak existuje PTS M' běžící v čase $10 \cdot p(|x|)$.*

Důkaz. PTS M spustí M, přičemž si počítá počet kroků. Pokud se M zastaví, zastaví se i M' se stejnou odpovědí.

Jestliže počet kroků přesáhne $10 \cdot p(|x|)$, stroj M' odmítne. To ale nastane s pravděpodobností nejvýše $1/10$. Pokud M odpoví správně s pravděpodobností $2/3$, pak M' odpoví správně s pravděpodobností $2/3 - 1/10 > 0.55$ a díky redukci chyby víme, že to již stačí. □

Doteď jsme probírali oboustrannou chybu. Zaměřme se tedy na případy, kdy náhodnost nám umožňuje dostat pouze jednostrannou, nebo dokonce žádnou chybu. V druhém případě bude pravděpodobnost ovlivňovat pouze čas běhu.

Definice. $\text{RTIME}(T(n))$ je třída jazyků, pro které existuje PTS M pracující v (očekávaném) čase $T(n)$ takový, že pro $x \in M : \Pr[M(x) = 1] \geq 2/3$ a pro $x \notin M : \Pr[M(x) = 0] = 1$.

Třída $\mathsf{RP} = \bigcup_{c>0} \text{RTIME}(n^c)$ (Randomized Polytime).

Definice. $\text{ZPTIME}(T(n))$ je třída jazyků, pro které existuje PTS M pracující v očekávaném čase $T(n)$ takový, že vždy odpoví správně.

Třída $\mathsf{ZPP} = \bigcup_{c>0} \text{ZPTIME}(n^c)$ (Zero-error Probabilistic Polytime).

Nyní si všimněme, že celkem triviálně platí tyto vztahy:

- $\mathsf{RP} \subseteq \mathsf{NP}$: Třída RP říká, že existuje alespoň konstantní poměr přijímajících větví, třídě NP stačí jedna.
- $\mathsf{BPP} = \mathsf{Co\text{-}BPP}$: Oboustranná chyba je pro obě odpovědi dostatečně vysoká, lze PTS tedy prohodit odpovědi.
- $\mathsf{RP} \subseteq \mathsf{BPP}$, $\mathsf{Co\text{-}RP} \subseteq \mathsf{BPP}$: Jednostranná chyba je silnější požadavek, než oboustranná.

Věta 5.5. $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{Co\text{-}RP}$.

Důkaz. Nechť $L \in \mathsf{ZPP}$. Pak $L \in \mathsf{RP}$ a $L \in \mathsf{Co\text{-}RP}$, přímo z definice.

Předpokládejme, že $L \in \mathsf{RP} \cap \mathsf{Co\text{-}RP}$. Pak existují PTS M a $\bar{M}$ takové: Když $x \in L$, pak $\Pr[M(x) = 1] \geq 2/3$ a $\Pr[\bar{M} = 1] = 1$. Naopak pro $x \notin L$ platí $\Pr[M(x) = 0] = 1$ a $\Pr[\bar{M} = 0] \geq 2/3$.

Tudíž, pokud $M(x) = 1$, pak $x \in L$ a jestliže $\bar{M}(x) = 0$, $x \notin L$. Na základě toho zkonstruujeme stroj M', který na vstupu x spustí M a $\bar{M}$ souběžně. Podle odpovědí se rozhodneme:

- $M(x) = 1$, pak vrátí $M'(x) = 1$.
- $\bar{M}(x) = 0$, pak vrátí $M'(x) = 0$.
- Jinak M' nerozhodne a spustí $M, \bar{M}$ znovu na x.

Jistě stroj M' odpoví zaručeně správně. Navíc, $\Pr[M(x) = 0 \wedge \bar{M}(x) = 1] \leq 1/3$, což lze rozdělit dle podmínky, že $X \in L$ a $x \notin L$.

Tudíž očekávaný počet opakování je nejvýše $\sum_{i=1}^{\infty} i \cdot (1/3)^i$, což lze odhadnout konstantou. Oba stroje $M, \bar{M}$ navíc pracují v polynomiálním čase. □

Věta 5.6 (Sipser-Gács)**.** $\mathsf{BPP} \subseteq \Sigma_2 \cap \Pi_2$.

Důkaz. Stačí ukázat $\mathsf{BPP} \subseteq \Sigma_2$. Nechť $L \in \mathsf{BPP}$ je libovolný. Díky větě o redukci chyby pro BPP existuje DTS M pracující v polynomiálním čase, který na vstupu délky n používá $m = \text{poly}(n)$ náhodných bitů. Pro něj platí pro $x \in L : \Pr[M(x, r) = 1] \geq 1 - 1/2^n$ a pro $x \notin L : \Pr[M(x, r) = 1] \leq 1/2^n$.

Pro každý vstup x označme $S_x = \{r \mid M(x, r) = 1\}$. Pak pro $x \in L$ platí $|S_x| \geq (1 - 1/2^n) \cdot 2^m$ a pro $x \notin L$ je $|S_x| \leq 1/2^n \cdot 2^m$.

Vhodný počet „náhodných posunutí“ S_x v případě $x \in L$ pokryje všechna $\{0,1\}^m$, kdežto v případě $x \notin L$ stejný počet posunutí S_x nemůže pokrýt $\{0,1\}^m$ pro žádný výběr posunutí.

Pro $r, u \in \{0,1\}^m$ budeme uvažovat $r \oplus u$ jako XOR. Pro $S \subseteq \{0,1\}^m$ budeme značit $S \oplus u = \{x \oplus u \mid x \in S\}$. Zvolme vhodný počet $k = \lceil m/n \rceil + 1$.

Lemma 5.7. *Nechť $S \subseteq \{0,1\}^m$ taková, že $|S| \leq 2^{m-n}$, nechť $u_1, \ldots, u_k \in \{0,1\}^m$. Pak $\bigcup_{i=1}^{k}(S \oplus u_i) \subsetneq \{0,1\}^m$.*

Důkaz. Pro každé i platí $|S \oplus U_i| = |S|$. Tím pádem $|\bigcup_i S \oplus U_i| \leq k \cdot |S| \leq (m/n+2) \cdot 2^{m-n} < 2^m$ pro dostatečně velká n. □

Lemma 5.8. *Nechť $S \subseteq \{0,1\}^m$ taková, že $|S| \geq (1-2^{-n}) \cdot 2^m$. Pak existují $u_1, \ldots, u_k \in \{0,1\}^m$ takové, že $\bigcup_{i=1}^{k}(S \oplus u_i) = \{0,1\}^m$.*

Důkaz. Stačí ukázat, že při náhodném výběru řetězců u_i platí, že $\Pr[\bigcup_i(S \oplus u_i) = \{0,1\}^m] > 0$, nebo ekvivalentně $\Pr[\bigcup_i(S \oplus u_i) \subsetneq \{0,1\}^m] < 1$. Tato pravděpodobnost je též $\Pr[\exists r \in \{0,1\}^m : r \notin \bigcup_i(S \oplus u_i)] < 1$.

Řetězec r je špatný pro i právě, když $r \notin S \oplus u_i$, nebo díky komutativitě $\oplus$ též $r \oplus u_i \notin S$. Pokud $u_1, \ldots, u_k$ byly vybrané uniformně nezávisle z $\{0,1\}^m$, tak také $\forall r : r \oplus u_i$ mají uniformní a nezávislou distribuci.

Pak $\forall r : \Pr[r \oplus u_i \in S] \geq 1 - 2^{-n}$ a opačný jev $\Pr[r \oplus u_i \notin S] \leq 2^{-n}$. Tudíž $\Pr[\forall i : r \oplus u_i \notin S] \leq (2^{-n})^k$, a tedy $\Pr[\exists r : r \notin \bigcup_i(S \oplus u_i)] \leq 2^m \cdot 2^{-nk}$.

Pro námi zvolené k dostáváme $2^m \cdot 2^{-nk} < 2^m \cdot 2^{-m} = 1$, jelikož $nk = n(\lceil m/n \rceil) + 1 > m$. □

Z těchto dvou lemmat platí, že $x \in L \Leftrightarrow \exists u_1, \ldots, u_k \in \{0,1\}^m \forall r \in \{0,1\}^m : r \in \bigcup_{i=1}^{k}(S_x \oplus u_i)$. To lze přepsat na $x \in L \Leftrightarrow \exists \forall \bigwedge_i (M(x, r \oplus u_i) = 1)$, a tato podmínka je v P. Použijeme definici a $L \in \Sigma_2$. □

Věta 5.9. BPP $\subseteq$ P/poly.

Důkaz. Nechť $L \in$ BPP je libovolný. Díky větě o redukci chyby pro BPP existuje DTS M pracující v polynomiálním čase, který na vstupu délky n používá $m = \text{poly}(n)$ náhodných bitů. Pro něj platí pro $\Pr[M(x) \neq L(x)] \leq 1/2^{n+1}$.

Řetězec náhodných bitů $r \in \{0,1\}^m$ je špatný pro $x \in \{0,1\}^n$, jestliže platí $M(x,r) \neq L(x)$. Pro pevné x existuje nejvýše $2^m/2^{n+1}$ špatných r. To znamená, že existuje nejvýše $2^n \cdot 2^m/2^{n+1} = 2^{m-1}$ řetězců r špatných vzhledem k x. Proto existuje $r_0 \in \{0,1\}^m$, které je dobré pro každé $x \in \{0,1\}^n$.

Z věty P $\subseteq$ P/poly víme, že DTS M se pro každou pevnou délku vstupu dá simulovat obvodem polynomiálně velkým vzhledem k délce vstupu a délce výpočtu M, tedy vše je poly(n). Pro vstupy délky $n+m$ simulujeme práci DTS M na obvodu C_{n+m} kde navíc u vstupu (x,r) zadrátujeme r napevno na r_0. Tudíž platí $\forall x \in \{0,1\}^n : M(x,r) = M(x,r_0) = L(x) = C_{n+m}(x)$. □

5.1 PCP věta a neaproximovatelnost

PCP znamená Probalistically Checkable Proofs.

NP-úplné problémy obvykle bývají rozhodovací verze optimalizačních problémů. Když si vezmeme takové optimalizační algoritmy, tak z hlediska řešitelnosti jsou si všechny rovny – když umíme polynomiálně vyřešit jeden, umíme převoditelností vyřešit všechny.

S aproximovatelností to je ale jiné. Batoh i TSP jsou NP-těžké. Pro batoh máme plně polynomiální aproximační schéma, zatímco existence aproximačního algoritmu s konstantním poměrem pro TSP by implikovalo $\mathsf{P} = \mathsf{NP}$. PCP věta se snaží tento koncept zobecnit a dodat systematický nástroj rozpoznávání aproximovatelnosti.

Příklad. Uvažme problém Max-3-SAT. Vstupem je 3-CNF φ a cílem je ohodnotit proměnné tak, abychom splnili co nejvíce klauzulí.

Řekněme, že val(φ) je maximální počet splnitelných formulí.

Definice. L je jazyk, $q, r : \mathbb{N} \to \mathbb{N}$. Řekněme, že L má $(r(n), q(n))$-PCP verifier, pokud existuje pravděpodovnostní algoritmus V běžící v polynomiálním čase s těmito vlastnostmi:

- (Efficiency) Na vstupu $x \in \{0,1\}^n$ a s náhodným přístupem k řetězci Π (důkaz) V použije $\leq r(n)$ náhodných bitů udělá $\leq q(n)$ dotazů na bity Π. Pak vydá 1 nebo 0 a tato náhodná verifikace se označuje $V^{\Pi}(x)$.
- (Correctness) $x \in L$, pak $\exists \Pi : \Pr[V^{\Pi}(x) = 1] = 1$ a takové Π nazýváme správným důkazem pro x.
- (Soundness) $x \notin L$, pak $\forall Pi : \Pr[V^{\Pi}(x) = 1] \leq 1/2$.

Říkáme, že $L \in \mathrm{PCP}(r(n), q(n))$, pokud existují konstanty $c, d > 0$ takové, že L má $(c \cdot r(n), d \cdot q(n))$-PCP verifier.

Řetězce Π stačí uvažovat délky $2^{\mathcal{O}(r(n))}$, protože pouze tolik „adres“ v důkazu Π lze vygenerovat s nenulovou pravděpodobností.

Lemma 5.10. $\mathrm{PCP}(r(n), q(n)) \subseteq \mathrm{NTIME}(2^{\mathcal{O}(r(n)) \cdot q(n))})$.

Důkaz. NTS M nejprve vygeneruje důkaz Π a pak deterministicky generuje všechny řetězce náhodných bitů délky $r(n)$ a nakonec simuluje běh verifikátoru. Pouze pokud všechny náhodné řetězce přijmou, tak M přijme. □

Důsledek 5.11. $\mathrm{PCP}(\log n, 1) \subseteq \mathrm{NTIME}(2^{\mathcal{O}(\log n)}) = \mathsf{NP}$.

Překvapivě, platí ještě silnější tvrzení:

Věta 5.12 (PCP-věta (klasická verze))**.** $\mathsf{NP} = \mathrm{PCP}(\log n, 1)$.

Příklad. Jazyk $\mathrm{GNI} = \{(G_1, G_2) \mid G_1 \not\equiv G_2\}$. Ukážeme, že $\mathrm{GNI} \in \mathrm{PCP}(p(n), 1)$.

Mějme G_1, G_2 na n vrcholech. Setřídíme všechny grafy na n vrcholech podle jejich „kódů“ a Π jsou řetězce indexované těmito kódy, $|\Pi| = 2^{\mathcal{O}(n^2)}$. Správný důkaz obsahuje na pozici odpovídající grafu H 0, pokud $H \equiv G_0, H \not\equiv G_1$; 1, pokud $H \equiv G_1, H \not\equiv G_0$; jinak náhodný bit.

Verifikátor náhodně vygeneruje $b \in \{0, 1\}$ a permutaci $[n]$. Nechť H je G_b po provedení příslušné permutaci vrcholů, pak se podívá do Π na pozici odpovídající H. Přijme příslušný bit, pokud je výsledek rovný b.

Věta 5.13 (PCP-věta (aproximační verze))**.** $\exists \rho < 1 : \forall L \in \mathsf{NP}$*, existuje polynomiálně vyčíslitelná funkce f mapující řetězce na reprezentaci 3CNF formulí tak, že:*

- $x \in L \Rightarrow \mathrm{val}(f(x)) = 1$,
- $x \notin L \Rightarrow \mathrm{val}(f(x)) < \rho$.

Chceme ukázat, že obě varianty PCP-věty jsou si ekvivalentní. K tomu ale musíme použít mezikrok.

Definice. qCSP φ je $\{\varphi_1, \ldots, \varphi_m\}$, kde $\varphi_i : \{0, 1\}^n \to \{0, 1\}$ je constraint, který závisí na nejvýše q proměnných. Ohodnocení $u \in \{0, 1\}^n$ splňuje φ_i právě, když $\varphi_i(u) = 1$.

Dále $\mathrm{val}(\varphi) = \max_u(\sum_i \varphi_i(u)/m)$. Řekneme, že φ je splintelná, pokud $\mathrm{val}(\varphi) = 1$.

Definice. ρ-gap-qCSP je problém zjistit pro dané qCSP, jestli:

- $\mathrm{val}(\varphi) = 1$ a jde o kladnou instanci, nebo
- $\mathrm{val}(\varphi) < \rho$ a jde o zápornou instanci.

Pokud existuje NP-těžký problém L a funkce f (polynomiálně vyčíslitelná) mapující vstupní řetězce L na instance ρ-gap-qCSP, pro kterou $x \in L \Rightarrow \mathrm{val}(f(x)) = 1$ a $x \notin L \Rightarrow \mathrm{val}(f(x)) < \rho$, řekněme, že ρ-gap-qCSP je NP-těžká.

Věta 5.14 (PCP-věta (CSP verze))**.** *Existují konstanty q, ρ takové, že ρ-gap-qCSP je NP-těžká.*

6 Ladnerova věta

Podívejme se na třídu NP. Pokud $\mathsf{P} \neq \mathsf{NP}$, pak jsou jistě NP-úplné problémy disjunktní s polynomiálními problémy. Existuje ale nějaký problém mezi P a $\mathsf{NP\acute{U}}$?

Věta 6.1 (Ladnerova)**.** $\mathsf{P} \subsetneq \mathsf{NP}$*, pak existuje* A *takové, že* $A \in \mathsf{NP}, A \notin \mathsf{NPÚ}, A \notin \mathsf{P}$*. Takovému problému se říká* NP*-intermediate.*

Důkaz. Uvažujme $M_1, M_2, \ldots$ očíslování DTS s orákulem pracujících v polynomiálním čase. BÚNO M_i pracuje v čase n^i, kde n je délka vstupu.

Pokud $X \in \mathsf{P}$, tak musí existovat i takové, že $X = L(M_i, \emptyset)$. Dále, když je $X \in \mathsf{NPÚ}$, pak existuje i takové, že $\mathrm{SAT} = L(M_i, X)$. Náš cíl je tedy konstrukce jazyka $A \in \mathsf{NP}$ takového, že $A \notin \{L(M_i, \emptyset)\}$ a $\mathrm{SAT} \notin \{L_{M_i,A}\}$.

Chceme explicitní konstrukci univerzálního NTS N, který rozpoznává A v polynomiálním čase pro pevný polynom p. N bude používat dva typy procedury AGREE:

- AGREE$(M_i^{\emptyset}, y)$: N simuluje M_i na vstupu y s orákulem $\emptyset$ a navíc testuje, zda $y \in A$. Když obojí dá stejnou odpověď, odpoví ANO, jinak NE.
- AGREE(M_i^{A}, y): N simuluje M_i na vstupu y s orákulem A a navíc testuje, zda $y \in \mathrm{SAT}$. Když obojí dá stejnou odpověď, odpoví ANO, jinak NE.

Pak práce N na vstupu $x \in \{0,1\}^n$ bude vypadat:

Nejprve nastav $i \leftarrow 1, y \leftarrow$ "0". Dále, Dokud N neudělá $p(n)$ kroků, tak dokola opakuj:

(a) Dokud AGREE$(M_i^{\emptyset}, y)$, tak vezmi lexikograficky následující y.

(b) Pak dokud AGREE(M_i^{A}, y), tak vezmi lexikograficky následující y.

(c) Nakonec inkrementuj y.

Pokud smyčka skončila v bodě (a), tak N přijme x právě, když $x \in \mathrm{SAT}$. Jinak N odmítne x.

Skrytý předpoklad je, že uvažujeme taková y, aby $|y| \leq \log |x|$. To děláme proto, aby $2^{|y|} \leq |x|$, takže při testování SAT (v orákulovém dotazu) můžeme v polynomiálním čase zkusit všechny možnosti. Stroj N se tedy chová jako DTS až na poslední test $x \in \mathrm{SAT}$.

Viditelně $A \in \mathsf{NP}$, protože $A \in L(N)$.

Sporem nyní předpokládejme, že $A \in \mathsf{NPÚ}$. Tedy $\exists i : \mathrm{SAT} = L(M_i, A)$ a nechť i je nejmenší takové. Tudíž $\forall j < i : \mathrm{SAT} \neq L(M_i, A)$. Navíc, $\mathsf{P} \neq \mathsf{NP}$ a $A \in \mathsf{NPÚ}$, proto $\forall j : A \neq L(M_i, \emptyset)$. Existuje $2i - 1$ řetězců y, které zajistí, že pro dostatečně dlouhé x stroj N uvázne v (b) pro stroj M_i a vždy odmítne. Tudíž A je konečný a $A \in \mathsf{P}$, spor.

Předpokládejme ještě sporem, že $A \in \mathsf{P}$. Tedy $\exists i : A = L(M_i, \emptyset)$ a nechť i je nejmenší takové. Tudíž $\forall j < i : A \neq L(M_i, \emptyset)$. Navíc, $P(\mathsf{P}) = \mathsf{P}$ a $\mathsf{NPÚ} \neq \mathsf{P}$,

takže $\forall j : \mathrm{SAT} \neq L(M_i, A)$. Tudíž existuje $2i - 1$ řetězců y, které zajistí, že pro dostatečně dlouhé x stroj N uvázne v (a) pro stroj M_i a od určité chvíle $A = \mathrm{SAT}$, tudíž $A \in \mathsf{NP\acute{U}}$, spor. □