Základy složitosti a vyčíslitelnosti

NTIN090

Petr Kučera

2024/25 (9. přednáška)

Dnešní plán

- Převod SAT na 3-SAT
- NP-úplnost Vrcholového pokrytí
- Seznam několika NP-úplných problémů
- Třída co-NP
- Třída #P

3-SAT

SAT (připomenutí)

Splnitelnost (SAT)

Instance: Formule φ v KNF.

Otázka: Je formule φ splnitelná?

Věta

SAT je NP*-úplný problém.*

3-SAT

3-KNF formule φ je v 3-KNF, pokud je v KNF a každá klauzule obsahuje právě 3 literály

3-SAT

Instance: Formule φ v 3-KNF.

Otázka: Je φ splnitelná?

Věta

Problém 3-SAT je NP-úplný.

NP-úplnost 3-SATu

3-SAT patří do třídy NP

- Týž polynomiální verifikátor jako pro SAT
- Ověřuje, jestli je daná formule φ splněna daným ohodnocením $\mathbf{a}$

3-SAT je NP-těžký

- SAT je polynomiálně převoditelný na 3-SAT

φ je splnitelná $\iff$ ψ je splnitelná

Převod SAT na 3-SAT

- Mějme KNF φ, jež má
  - n proměnných $x_1, \ldots, x_n$
  - m klauzulí $C_1, \ldots, C_m$
- Popíšeme konstrukci 3-KNF ψ, pro kterou platí

$$\varphi \text{ je splnitelná} \iff \psi \text{ je splnitelná}$$

- Pro každou klauzuli C_j, $j = 1, \ldots, m$
  - Podle potřeby přidáme nové proměnné
  - Sestrojíme konjunkci nových klauzulí α_j
  - Ohodnocení splňující C_j může být rozšířeno na model α_j
  - Ohodnocení splňující α_j splňuje C_j
- Definujeme $\psi = \bigwedge_{j=1}^{m} \alpha_j$
- Rozlišíme několik případů dle velikosti klauzule C_j

Prázdná klauzule

$C_j = \bot$ (prázdná klauzule)

- C_j není splnitelná $\implies$ φ je nesplnitelná
- α_j je konjunkcí všech klauzulí délky 3 na nových proměnných y_1, y_2 a y_3

$$\begin{aligned}\alpha_j = {} & (y_1 \vee y_2 \vee y_3) \wedge (y_1 \vee y_2 \vee \neg y_3) \\ & \wedge (y_1 \vee \neg y_2 \vee y_3) \wedge (y_1 \vee \neg y_2 \vee \neg y_3) \\ & \wedge (\neg y_1 \vee y_2 \vee y_3) \wedge (\neg y_1 \vee y_2 \vee \neg y_3) \\ & \wedge (\neg y_1 \vee \neg y_2 \vee y_3) \wedge (\neg y_1 \vee \neg y_2 \vee \neg y_3)\end{aligned}$$

C_j ani α_j nemají model.

Klauzule délky 1

$C_j = l$ pro nějaký literál l

- Přidáme dvě nové proměnné y_1 a y_2

$$\alpha_j = (l \vee y_1 \vee y_2) \wedge (l \vee y_1 \vee \neg y_2) \\ \wedge (l \vee \neg y_1 \vee y_2) \wedge (l \vee \neg y_1 \vee \neg y_2)$$

Nechť $\mathbf{a}$ je ohodnocení, které přiřazuje hodnotu l, y_1 a y_2.

$\mathbf{a}$ splňuje $C_j \iff \mathbf{a}$ splňuje α_j

Klauzule délky 2

$C_j = l_1 \vee l_2$ pro nějaké literály l_1 a l_2

- Přidáme novou proměnnou y

$$\alpha_j = (l_1 \vee l_2 \vee y) \wedge (l_1 \vee l_2 \vee \neg y)$$

Nechť $\mathbf{a}$ je ohodnocení, které přiřazuje hodnotu l_1, l_2 a y.

$\mathbf{a}$ splňuje $C_j \iff \mathbf{a}$ splňuje α_j

Klauzule délky 3

$C_j = l_1 \vee l_2 \vee l_3$ pro nějaké literály l_1, l_2 a l_3

- Ponecháme C_j beze změny

$$\alpha_j = C_j$$

Nechť $\mathbf{a}$ je ohodnocení, které přiřazuje hodnotu l_1, l_2 a l_3.

$\mathbf{a}$ splňuje $C_j \iff \mathbf{a}$ splňuje α_j

Klauzule velikosti $k > 3$ (myšlenka)

$C_j = l_1 \vee \cdots \vee l_k$ pro $k > 3$ a nějaké literály $l_1, \ldots, l_k$

Myšlenka:

- Přidáme novou proměnnou y a položíme

$$\beta = (l_1 \vee l_2 \vee y) \wedge (\neg y \vee l_3 \vee \cdots \vee l_k)$$

- Ohodnocení $\mathbf{a}'$, které splňuje β, splňuje také C_j
- Pokud $\mathbf{a}$ splňuje C_j
  - Můžeme rozšířit $\mathbf{a}$ na ohodnocení $\mathbf{a}'$, které splňuje β
  - Stačí zvolit vhodnou hodnotu y
- Dělení opakujeme, dokud nemáme jen klauzule velikosti 3

Klauzule velikosti $k > 3$

$C_j = l_1 \vee \cdots \vee l_k$ pro $k > 3$ a nějaké literály $l_1, \ldots, l_k$

- Přidáme nové proměnné $y_1, \ldots, y_{k-3}$

$$\alpha_j = (l_1 \vee l_2 \vee y_1) \wedge (\neg y_1 \vee l_3 \vee y_2) \wedge \cdots \wedge (\neg y_{i-2} \vee l_i \vee y_{i-1})$$
$$\wedge \cdots \wedge (\neg y_{k-4} \vee l_{k-2} \vee y_{k-3}) \wedge (\neg y_{k-3} \vee l_{k-1} \vee l_k)$$

Ukážeme, že

1. pokud ohodnocení $\mathbf{a}'$ splňuje α_j, pak splňuje i C_j
2. pokud $\mathbf{a}$ přiřazuje hodnoty literálům $l_1, \ldots, l_k$ a pokud $\mathbf{a}$ splňuje C_j, pak jej lze rozšířit na model α_j

Model α_j splňuje C_j

$$\alpha_j = (l_1 \vee l_2 \vee y_1) \wedge (\neg y_1 \vee l_3 \vee y_2) \wedge \cdots \wedge (\neg y_{i-2} \vee l_i \vee y_{i-1}) \\ \wedge \cdots \wedge (\neg y_{k-4} \vee l_{k-2} \vee y_{k-3}) \wedge (\neg y_{k-3} \vee l_{k-1} \vee l_k)$$

- Uvažme situaci, kde všechny literály l_i mají hodnotu 0
- Obdržíme formuli

$$\alpha'_j = y_1 \wedge (\neg y_1 \vee y_2) \wedge \cdots \wedge (\neg y_{i-2} \vee y_{i-1}) \wedge \cdots \wedge (\neg y_{k-4} \vee y_{k-3}) \wedge \neg y_{k-3}$$

- Dostáváme, že $\alpha'_j \models y_1$, tedy

$$\alpha'_j \equiv y_1 \wedge y_2 \wedge (\neg y_2 \vee y_3) \wedge \cdots \wedge (\neg y_{i-2} \vee y_{i-1}) \wedge \cdots \wedge (\neg y_{k-4} \vee y_{k-3}) \wedge \neg y_{k-3}$$

Model α_j splňuje C_j

$$\alpha'_j \equiv y_1 \wedge y_2 \wedge (\neg y_2 \vee y_3) \wedge \cdots \wedge (\neg y_{i-2} \vee y_{i-1}) \wedge \cdots \wedge (\neg y_{k-4} \vee y_{k-3}) \wedge \neg y_{k-3}$$

- Platí $\alpha'_j \models y_2$
- Indukcí odvodíme $\alpha'_j \models y_{k-3}$
- Navíc $\alpha'_j \models \neg y_{k-3}$
- Dohromady tedy $\alpha'_j \models \bot$
- Jinými slovy, α'_j je nesplnitelná
- Každý model α_j musí splnit nějaký z literálů $l_1, \ldots, l_k$

Každý model α_j splňuje C_j

Modely C_j lze rozšířit na modely α_j

- Nechť $\mathbf{a}$ je model C_j
  - $\mathbf{a}$ splňuje některý z literálů $l_1, \ldots, l_k$
  - Označme p index některého splněného literálu
  - Tedy $\mathbf{a}(l_p) = 1$
- Popíšeme model $\mathbf{a}'$ formule α_j, který rozšiřuje $\mathbf{a}$
  - $\mathbf{a}'(x_i) = \mathbf{a}(x_i)$, $i = 1, \ldots, n$
  - Určíme navíc hodnoty proměnných $y_1, \ldots, y_{k-3}$

Rozšíření modelu C_j

- Pro $i = 1, \ldots, k-3$, položíme $\mathbf{a}'(y_i) = \begin{cases} 1 & i \leq p-2 \\ 0 & i > p-2 \end{cases}$
- α_j je potom splněná ohodnocením $\mathbf{a}'$

$$\alpha_j = (l_1 \vee l_2 \vee y_1) \wedge (\neg y_1 \vee l_3 \vee y_2) \wedge \cdots$$
$$\wedge (\neg y_{p-3} \vee l_{p-1} \vee y_{p-2}) \wedge (\neg y_{p-2} \vee l_p \vee y_{p-1})$$
$$\wedge (\neg y_{p-1} \vee l_{p+1} \vee y_p) \wedge \cdots$$
$$\wedge (\neg y_{k-4} \vee l_{k-2} \vee y_{k-3}) \wedge (\neg y_{k-3} \vee l_{k-1} \vee l_k)$$

Každý model C_j lze rozšířit na model α_j vhodným ohodnocením proměnných $y_1, \ldots, y_{k-3}$

Vlastnosti konstrukce

- ψ lze sestrojit v polynomiálním čase pro danou KNF φ
- Je-li $\mathbf{a}$ modelem φ

  $\Longrightarrow$ $\mathbf{a}$ splňuje všechny klauzule C_j

  $\Longrightarrow$ $\mathbf{a}$ lze rozšířit na model α_j, $j = 1, \ldots, m$

  $\Longrightarrow$ Rozšíření jsou vzájemně nezávislá

  $\Longrightarrow$ $\mathbf{a}$ lze rozšířit na model ψ
- Je-li $\mathbf{a}'$ model ψ

  $\Longrightarrow$ $\mathbf{a}'$ splňuje všechny podformule α_j

  $\Longrightarrow$ $\mathbf{a}'$ splňuje všechny klauzule C_j

  $\Longrightarrow$ $\mathbf{a}'$ je modelem φ

φ je splnitelná $\iff$ ψ je splnitelná.

Vrcholové pokrytí a další problémy

NP-úplnost Vrcholového pokrytí

Vrcholové pokrytí

Instance: Neorientovaný graf $G = (V, E)$ a celé číslo $k \geq 0$.

Otázka: Existuje množina vrcholů $S \subseteq V$ velikosti nejvýš k, která obsahuje alespoň jeden vrchol z každé hrany $\{u, v\} \in E$ (tedy $\{u, v\} \cap S \neq \emptyset$)?

Věta

3-SAT je polynomiálně převoditelný na Vrcholové pokrytí.

- Vrcholové pokrytí patří do NP
- Polynomiální verifikátor ověřuje, zda množina vrcholů S tvoří vrcholové pokrytí velikosti nejvýš k

Důsledek

Vrcholové pokrytí je NP-úplné.

Problémy související s Vrcholovým pokrytím

Klika (Clique) Obsahuje G jako podgraf kliku, tj. úplný graf, na k vrcholech?

Nezávislá množina (Independent Set): Obsahuje G nezávislou množinu velikosti k?

- Množina vrcholů S je nezávislá, pokud mezi žádnými dvěma vrcholy z S nevede hrana.

Hranového pokrytí (Edge Cover) Existuje v G množina hran velikosti nejvýš k, která pokrývá všechny vrcholy?

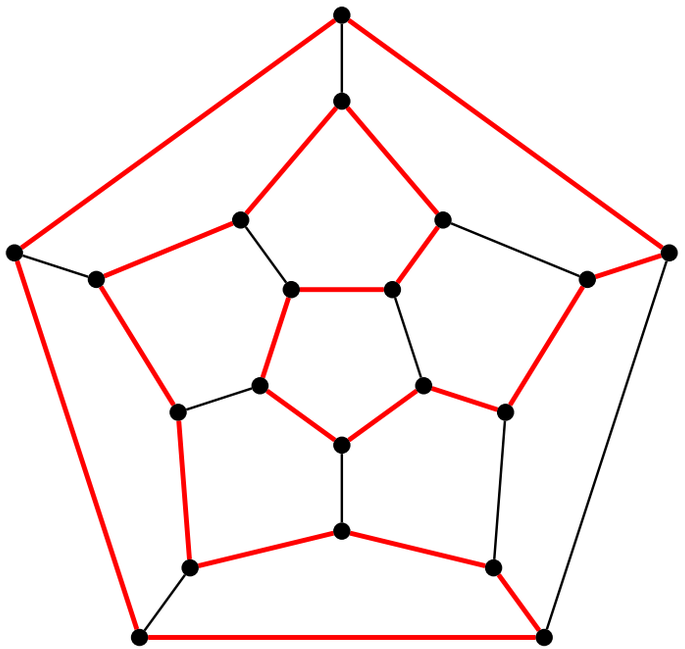
- Problémy Klika a Nezávislá množina jsou NP-úplné
- Problém Hranového pokrytí je řešitelný v polynomiálním čase

Hamiltonovská kružnice

Hamiltonovská kružnice (HK, Hamiltonian cycle)

Instance: Neorientovaný graf $G = (V, E)$.

Otázka: Existuje v grafu G cyklus vedoucí všemi vrcholy?



Věta (Bez důkazu)

Hamiltonovská kružnice je NP*-úplný problém.*

Třírozměrné párování

Třírozměrné párování (3DM, 3D Matching)

Instance: Množina $M \subseteq W \times X \times Y$, kde W, X a Y jsou množiny velikosti q.

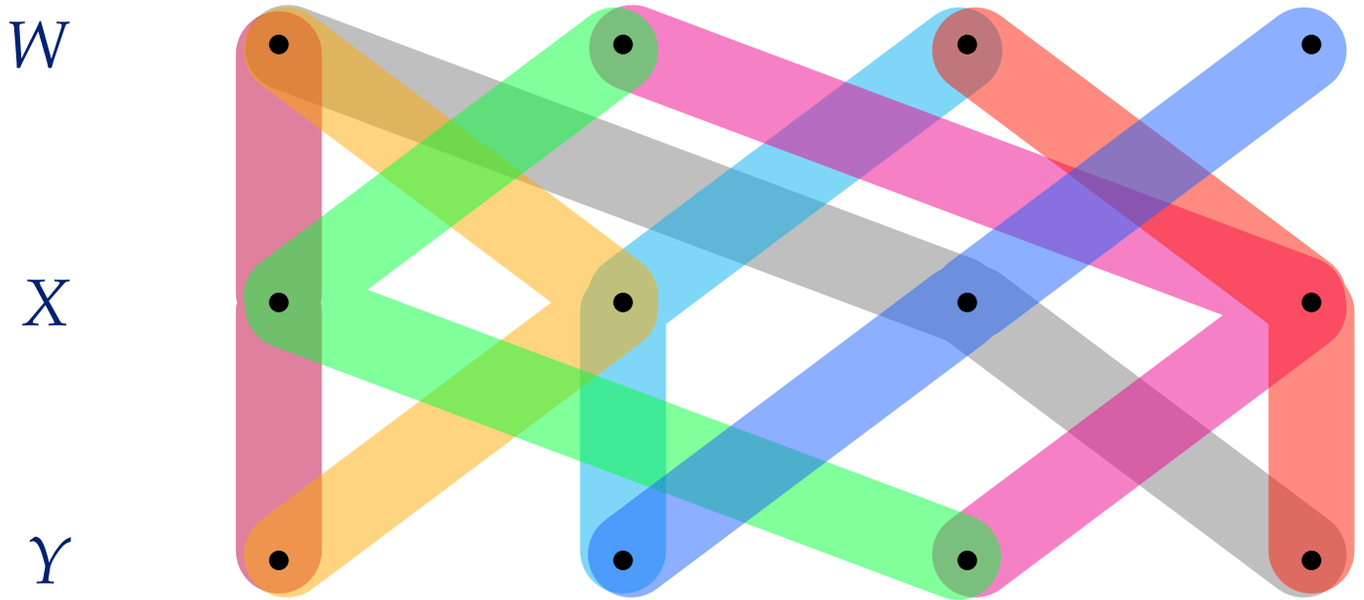
Otázka: Má M perfektní párování? Tj. lze v M vybrat q po dvou disjunktních trojic?

Věta (Bez důkazu)

Třírozměrné párování je NP-úplný problém.

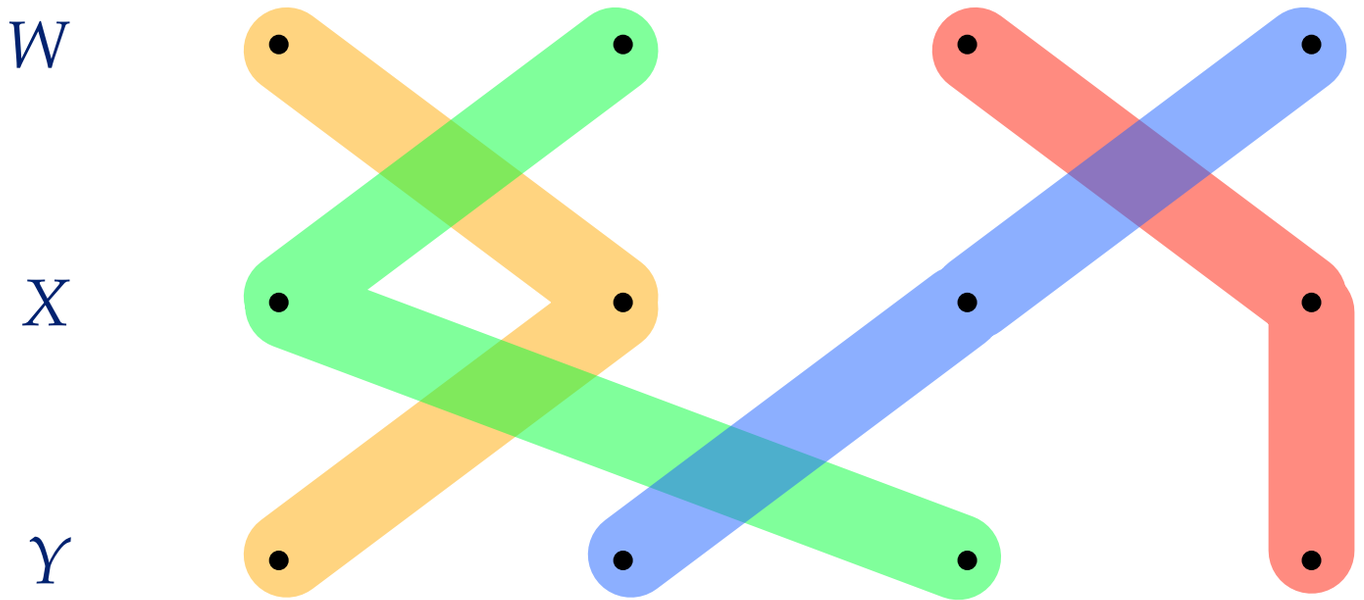
Příklad 3DM

Množina trojic $M \subseteq W \times X \times Y$



Příklad 3DM

Perfektní párování $M' \subseteq M$



Loupežníci

Loupežníci (Partition)

Instance: Množina předmětů A, s každým předmětem $a \in A$ asociované přirozené číslo $s(a)$ (váha, cena, velikost).

Otázka: Existuje $A' \subseteq A$, pro kterou platí, že

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

Věta (Bez důkazu)

Loupežníci je NP*-úplný problém.*

Batoh

Batoh (Knapsack)

Instance: Množina předmětů A, s každým předmětem $a \in A$ asociovaná velikost $s(a) \in \mathbb{N}$ a cena $v(a) \in \mathbb{N}$, velikost batohu $B \in \mathbb{N}$ a limit na cenu $K \in \mathbb{N}$.

Otázka: Lze vybrat množinu předmětů $A' \subseteq A$ tak, aby platilo

$$\sum_{a \in A'} s(a) \leq B \text{ a } \sum_{a \in A'} v(a) \geq K?$$

Věta

Batoh je NP-úplný problém.

- NP-těžkost lze ukázat snadným převodem z Loupežníků.

Rozvrhování

Rozvrhování (Scheduling)

Instance: Množina úloh U, s každou úlohou $u \in U$ asociovaná doba zpracování $d(u) \in \mathbb{N}$, počet procesorů m, limit $D \in \mathbb{N}$.

Otázka: Lze úlohy U rozdělit na m procesorů tak, aby byly všechny úlohy zpracované v časovém limitu D?

Věta

Rozvrhování je NP-úplný problém.

- NP-těžkost lze ukázat snadným převodem z Loupežníků.

Třída co-NP

Tautologie

Tautologie výroková formule, která je splněna každým ohodnocením

Příklady tautologií

Sylogismus $\big[(x \implies y) \wedge (y \implies z)\big] \implies (x \implies z)$

Kontrapozice $(x \implies y) \iff (\neg y \implies \neg x)$

De Morganova pravidla $\neg(x \vee y) \iff (\neg x \wedge \neg y)$

Důkaz sporem $[(\neg x \implies y) \wedge (\neg x \implies \neg y)] \implies x$

Problém tautologie

Tautologie (TAUT)

Instance: Výroková formule φ

Otázka: Je φ tautologie?

Patří TAUT do NP?

- Jak ověřit, zda daná formule je tautologie?
  - Zkontrolovat pravdivostní tabulku
  - Důkaz z axiomů výrokové logiky
  - ...

Není znám polynomiální verifikátor pro TAUT.

TAUT jako jazyk

$$\mathrm{TAUT} = \{\langle\varphi\rangle \mid \varphi \text{ je tautologie}\}$$

- Doplněk TAUT je definován takto

$$\overline{\mathrm{TAUT}} = \{\langle\varphi\rangle \mid \varphi \text{ není tautologie}\}$$

~ > existuje nějaké záporné ohodnocení

- φ není tautulogie $\iff$ $\varphi(\mathbf{a}) = 0$ pro nějaké ohodnocení $\mathbf{a}$
- Polynomiální verifikátor $V(\varphi, \mathbf{a})$ pro $\overline{\mathrm{TAUT}}$ ověří, zda $\varphi(\mathbf{a}) = 0$

$\overline{\mathrm{TAUT}}$ patří do NP

Třída co-NP

Definice

$$\text{co-NP} = \{A \mid \overline{A} \in \text{NP}\}$$

- Jazyk A patří do co-NP, právě když existuje polynomiální verifikátor V, pro nějž platí

$$A = \left\{x \mid (\forall y \in \{0,1\}^*) \left[V(x,y) \text{ přijme}\right]\right\}.$$

TAUT patří do co-NP

to je inverz verifikátoru NP, místo dříhozu platnosti jde o dříhoz spor.

co-NP-úplnost

Definice

Jazyk B je

→ "je alespoň tak těžký"

co-NP-těžký pokud pro každý jazyk $A \in \text{co-NP}$ platí $A \leq_m^P B$

co-NP-úplný pokud je co-NP-těžký a současně $B \in \text{co-NP}$

Věta

Jazyk A je co-NP-úplný, právě když $\overline{A}$ je NP-úplný

Důkaz.

Plyne z faktu, že pro každé dva jazyky A a B

$$A \leq_m^P B \iff \overline{A} \leq_m^P \overline{B}$$

□

Dva co-NP-úplné problémy

- Jazyky splnitelných a nesplnitelných formulí

$$\mathrm{SAT} = \{\langle\varphi\rangle \mid \varphi \text{ je splnitelná}\}$$
$$\overline{\mathrm{SAT}} = \{\langle\varphi\rangle \mid \varphi \text{ je nesplnitelná}\}$$

- Problém SAT je NP-úplný
- Z toho plyne, že $\overline{\mathrm{SAT}}$ je co-NP-úplný
- $\overline{\mathrm{SAT}} \leq_m^P \mathrm{TAUT}$ funkcí $f(\langle\varphi\rangle) = \langle\neg\varphi\rangle$

Problémy $\overline{\mathrm{SAT}}$ a TAUT jsou co-NP-úplné.

NP vs. co-NP

Věta

Pokud nějaký NP*-úplný problém* A *patří do* $\mathrm{co\text{-}NP}$*, pak* $\mathrm{NP} = \mathrm{co\text{-}NP}$*.*

- Protože $A \in \mathrm{co\text{-}NP}$, platí $\overline{A} \in \mathrm{NP}$
- Uvážíme-li $B \in \mathrm{NP}$, pak $B \leq_m^P A$ (z NP-úplnosti A)

$\Longrightarrow$ $\overline{B} \leq_m^P \overline{A}$, tedy $\overline{B} \in \mathrm{NP}$

$\Longrightarrow$ $B \in \mathrm{co\text{-}NP}$

- Z toho plyne $\mathrm{NP} \subseteq \mathrm{co\text{-}NP}$
- Díky symetrii též $\mathrm{co\text{-}NP} \subseteq \mathrm{NP}$

Nevíme, zda platí $\mathrm{NP} = \mathrm{co\text{-}NP}$ nebo $\mathrm{NP} \neq \mathrm{co\text{-}NP}$

P a rozhodnutelnost

Třída DEC

Jazyky rozhodnutelné Turingovými stroji

Třída P

Jazyky rozhodnutelné Turingovými stroji v polynomiálním čase

NP a částečná rozhodnutelnost

Třída PD (částečně rozhodnutelné)

- Přijímané Turingovými stroji
- Ověřitelné jazyky
- $A \in \mathrm{PD} \iff$ pro nějaký $B \in \mathrm{DEC}$

$$A = \{x \mid (\exists y \in \{0,1\}^*)[\langle x, y\rangle \in B]\}$$

Třída NP (nedeterministicky polynomiální)

- Přijímané nedeterministickými TS v polynomiálním čase
- Polynomiálně ověřitelné jazyky
- $A \in \mathrm{NP} \iff$ pro nějaký $B \in \mathrm{P}$ a polynom $p(n)$

$$A = \{x \mid (\exists y \in \{0,1\}^{p(|x|)})[\langle x, y\rangle \in B]\}$$

co-NP a částečná rozhodnutelnost

Třída co-PD

- Doplňky částečně rozhodnutelných jazyků
- $A \in \text{co-PD} \iff$ pro nějaký $B \in \text{DEC}$

$$A = \{x \mid (\forall y \in \{0,1\}^*)[\langle x, y \rangle \in B]\}$$

změna kvantifikátorů (pouze)

Třída co-NP

- Doplňky jazyků v NP
- $A \in \text{co-NP} \iff$ pro nějaký $B \in \text{P}$ a polynom $p(n)$

$$A = \{x \mid (\forall y \in \{0,1\}^{p(|x|)})[\langle x, y \rangle \in B]\}$$

Souvislost s Postovou větou

Dle Postovy věty

$$\mathrm{DEC} = \mathrm{PD} \cap \text{co-PD}$$

zatímco víme pouze

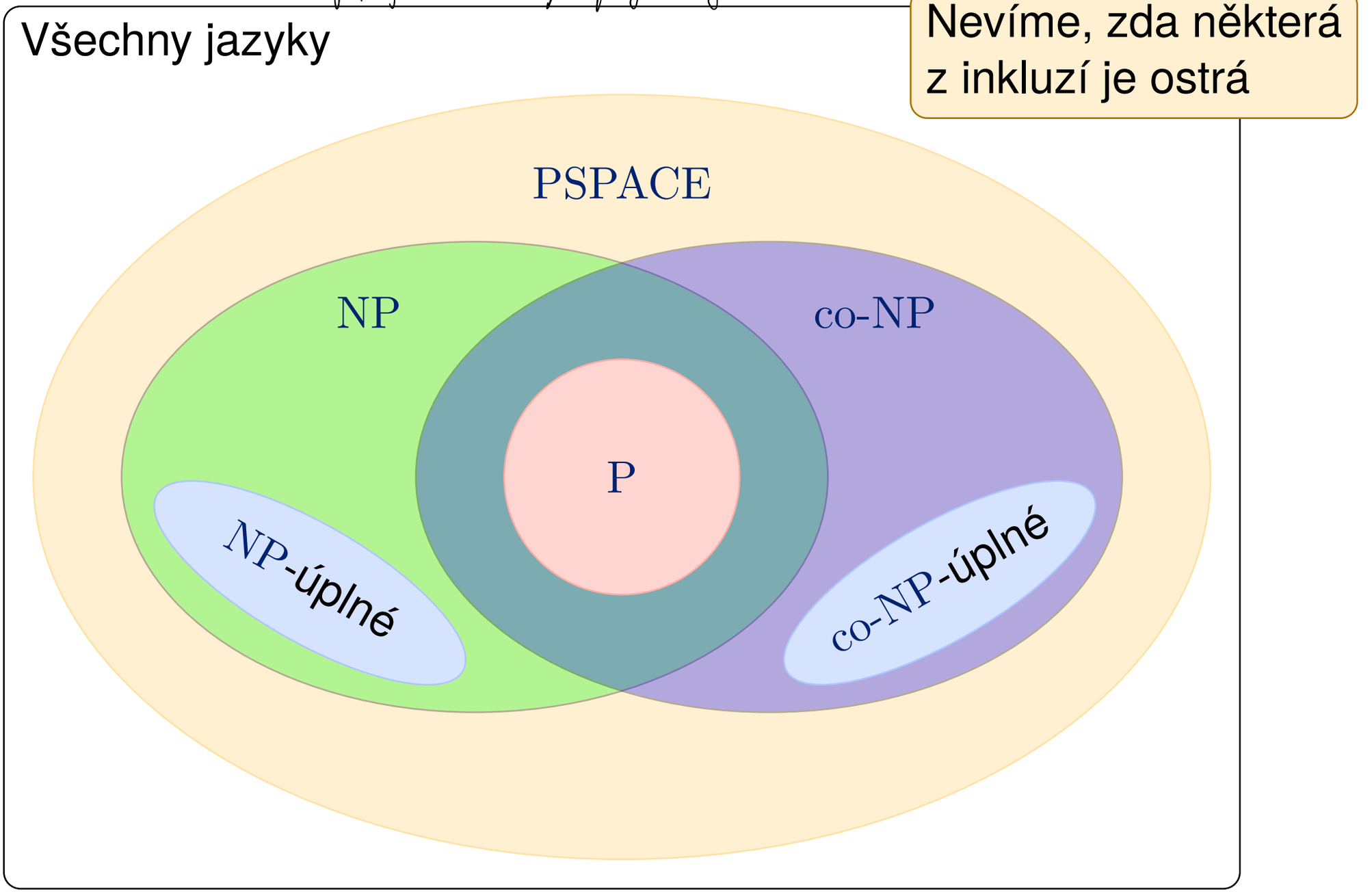
$$\mathrm{P} \subseteq \mathrm{NP} \cap \text{co-NP}$$

- Předpokládá se, že inkluze je ostrá
- Nicméně, neumíme vyloučit, že $\mathrm{P} = \mathrm{NP} = \text{co-NP}$

Vztahy mezi třídami

P = NP : klíčové je, jestli existuje polyn. alg. na řešení SAT u.

Nevíme, zda některá z inkluzí je ostrá



Třída #P

Počítání modelů

#SAT
Instance: Formule φ v KNF
Hodnota: Počet modelů φ — > Mohli bychom tím řešit jestli má (vůbec) nějaké řešení

Pokud $\#\mathrm{SAT}$ lze vyčíslit v polynomiálním čase, pak $\mathrm{P} = \mathrm{NP}$

- Polynomiální algoritmus pro $\#\mathrm{SAT}$ bychom mohli použít k rozhodnutí SAT

$$\varphi \text{ je splnitelná} \iff \#\mathrm{SAT}(\varphi) > 0$$

- Těžkost $\#\mathrm{SAT}$ je způsobená těžkostí SAT

Počítání cyklů

Těch cyklů může být exp. mnoho / i když umím 2^n různých počítat pouze do n-bitů, je to příliš těžké

#CYCLE

Instance: Orientovaný graf $G = (V, E)$

Hodnota: Počet cyklů G

- Snadné ověřit, zda G obsahuje cyklus
- Těžké určit počet cyklů v G

Věta

Pokud #CYCLE *lze vyčíslit v polynomiálním čase, pak* P = NP

- Ukážeme, že algoritmus vyčíslující #CYCLE lze použít k rozhodnutí problému Hamiltonovské kružnice
- Problém Hamiltonovské kružnice NP-úplný

Orientovaná Hamiltonovská kružnice

Orientovaná Hamiltonovská kružnice (OHK)

Instance: Orientovaný graf $G = (V, E)$.

Otázka: Obsahuje G cyklus délky $n = |V|$?

Věta (Bez důkazu)

Problém OHK je NP*-úplný.*

Rozhodnutí OHK počítáním cyklů

- Předpokládejme, že $\#\mathrm{CYCLE}$ lze vyčíslit v polynomiálním čase
- Ukážeme, že OHK lze rozhodnout v polynomiálním čase
- Popíšeme polynomiální převod orientovaného grafu G na orientovaný graf G'

$G = (V, E)$ má Hamiltonovskou kružnici $\iff$ $\#\mathrm{CYCLE}(G') \geq n^{n^2}$ kde $n = |V|$

Počet vrcholů G

Předpoklad: $n = 2^k$ pro nějaké $k \in \mathbb{N}$

V opačném případě upravíme G takto:

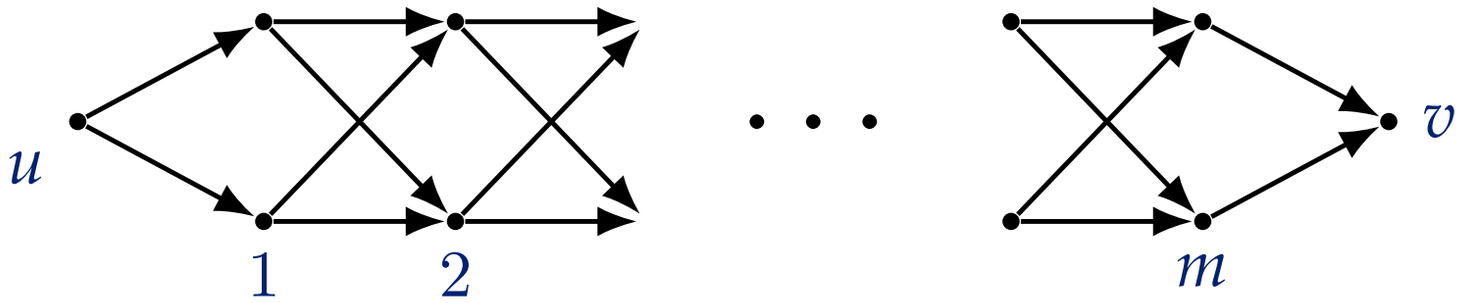
- Položme $k = \lceil \log_2 n \rceil$
- Vybereme vrchol $s \in V$
- Nahradíme s dvěma vrcholy s_{in} a s_{out} s hranami

$$E_{\text{in}} = \{(u, s_{\text{in}}) \mid (u, s) \in E\}$$
$$E_{\text{out}} = \{(s_{\text{out}}, u) \mid (s, u) \in E\}$$

- Spojíme s_{in} s s_{out} cestou procházející $2^k - n - 1$ nově přidanými vrcholy
- Výsledný graf
  - má 2^k vrcholů a
  - obsahuje hamiltonovskou kružnici, právě když ji obsahuje G

Konstrukce G'

Každou hranu (u, v) nahradíme následujícím podgrafem



- Podgraf má $m = n \log_2 n$ úrovní
- Podgraf je acyklický
  - Cykly G' odpovídají cyklům G
- Podgraf obsahuje 2^m orientovaných cest z u do v

v každém kroku mám na výběr, jestli chci přeskočit na dalším stupni nebo ne.

každou cestou reprezentuje 2^m hranami.

Cyklus v G délky ℓ dává vzniknout $(2^m)^\ell$ cyklům v G'.

Použití G' pro rozhodnutí OHK

- Nechť G obsahuje hamiltonovskou kružnici
  - Počet cyklů v G' je alespoň

$$(2^m)^n = (2^{n \log_2 n})^n = n^{n^2}$$

- Nechť G neobsahuje hamiltonovskou kružnici
  - Každý cyklus v G má délku nejvýš $n-1$
  - G obsahuje nejvýš n^{n-1} cyklů → v každém bodě cyklu si můžu vybrat příští vrchol.
  - Počet cyklů v G' je tedy nejvýš

$$(2^m)^{n-1} \cdot n^{n-1} = \left(2^{n \log_2 n} \cdot n\right)^{n-1} = n^{(n+1)(n-1)} = n^{n^2-1} < n^{n^2}$$

Zařídím to tak, že pokud tam ten cyklus má být, tak tak hrubě ten graf nafouknu, } Tím pak umím
že bez cyklu by to takhle nafouknout (ve smyslu počtu cyklů) nikdy nešlo. } rozhodnout jednoznačně existenci.

G obsahuje hamiltonovskou kružnici $\iff \#\mathrm{CYCLE}(G') \geq n^{n^2}$.

Třída #P

Definice

Funkce $f : \Sigma^* \to \mathbb{N}$ patří do třídy #P, pokud existuje polynomiální verifikátor V a polynom $p(n)$ takové, že pro každý $x \in \Sigma^*$ platí

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid V(x,y) \text{ přijme}\}|.$$

Alternativně

$$f(x) = \text{počet přijímajících výpočtů } M(x)$$

pro nějaký NTS M, který pracuje v polynomiálním čase.

Třídy NP a #P

Třída NP ptáme se po existenci polynomiálního certifikátu

- Má úloha řešení?

Třída #P zajímá nás počet polynomiálních certifikátů

- Kolik řešení úloha má?

Počítání certifikátů může být těžší než hledání jednoho

Příklad

Uvažme rozdíl mezi

- ověřováním acykličnosti orientovaného grafu (lehké) a
- počítáním cyklů v orientovaného grafu (těžké)

Pozitivita funkce f

Pozitivita f

Instance: $x \in \Sigma^*$

Otázka: $f(x) > 0$?

Věta

Je-li $f \in \#\mathrm{P}$, pak Pozitivita f patří do NP.

- Nechť V je polynomiální verifikátor a $p(n)$ polynom, které splňují

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid V(x,y) \text{ přijme}\}|$$

- Pak V je polynomiální verifikátor pro Pozitivita f
- Pozitivita f tedy patří do NP.

Počítání pomocí rozhodování

Věta

Pro funkci $f \in \#\mathrm{P}$ je otázka náležení do množiny $S_f = \{(x, N) \mid f(x) \geq N\}$ výpočetně ekvivalentní (až na polynomiální faktor) výpočtu f.

1. $(x, N) \in S_f$ lze jednoduše rozhodnout se znalostí hodnoty $f(x)$
2. $f(x)$ lze určit pomocí polynomiálního počtu dotazů typu „$(x, N) \in S_f$?“
   - Nechť V je polynomiální verifikátor a $p(n)$ polynom, které splňují

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid V(x, y) \text{ přijme}\}|$$

   - Tedy $0 \leq f(x) \leq 2^{p(|x|)}$
   - Použijeme binární vyhledávání k určení hodnoty $f(x)$
   - Stačí $O(p(|x|))$ dotazů

Prostorová složitost $\#\mathrm{P}$

Věta

Hodnotu $f \in \#\mathrm{P}$ lze vyčíslit v polynomiálním prostoru

- Nechť V je polynomiální verifikátor a $p(n)$ polynom, které splňují

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid V(x,y) \text{ přijme}\}|$$

- K určení hodnoty $f(x)$ stačí pustit $V(x,y)$ pro každý řetězec $y \in \{0,1\}^{p(|x|)}$
- $f(x)$ je pak počet přijatých certifikátů

Polynomiální převoditelnost funkcí

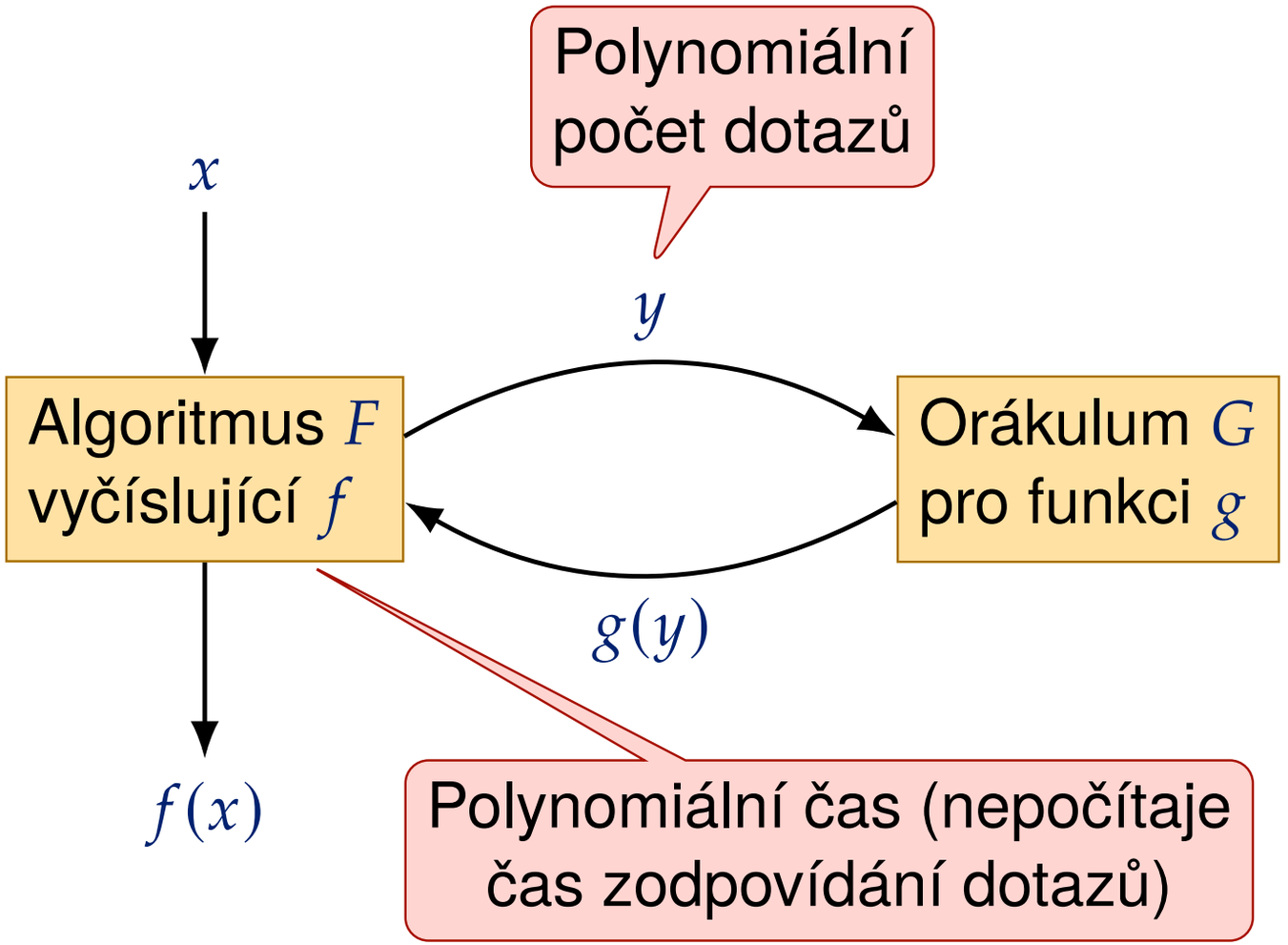
Definice

Funkce f je polynomiálně převoditelná na funkci g ($f \leq_P g$), pokud lze f vyčíslit v polynomiálním čase algoritmem, který má přístup k orákulu funkce g.

- Algoritmus vyčíslující $f(x)$
  - Má přístup k orákulu, které určí $g(y)$ pro libovolný řetězce y
  - Orákulu může položit polynomiální počet dotazů
  - Algoritmus pracuje v polynomiálním čase, za předpokladu, že orákulum odpovídá okamžitě

Převoditelnost funkcí (princip)

$f \leq_P g$ = „pomocí g umíme spočítat f v polynomiálním čase“



$\#\mathrm{P}$-úplnost

Definice

Funkce $g \in \#\mathrm{P}$ je $\#\mathrm{P}$-úplná, pokud $f \leq_P g$ pro každou funkci $f \in \#\mathrm{P}$.

Věta

Funkce $\#\mathrm{SAT}$ je $\#\mathrm{P}$-úplná.

- Plyne z převodu, kterým jsme ukazovali NP-úplnost SAT
- Podívejme se na to podrobněji

#P-úplnost #SAT

- Uvažme $f \in \#\mathrm{P}$
- Nechť V je polynomiální verifikátor a $p(n)$ polynom, které splňují

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} \mid V(x,y) \text{ přijme}\}|$$

- Uvažme následující nedeterministický TS M

Výpočet M se vstupem x

---

1. Nedeterministicky vyber $y \in \{0,1\}^{p(|x|)}$
2. Pusť $V(x,y)$
3. **if** $V(x,y)$ přijal **then** přijmi **else** odmítni

$f(x) =$ počet přijímajících výpočtů $M(x)$

Vyčíslení f pomocí #SAT

- V důkazu Cookovy-Levinovy věty jsme popsali konstrukci KNF φ, která s každým hodnocením $\mathbf{a}$ splňuje

$$\varphi(\mathbf{a}) = 1 \iff \mathbf{a} \text{ reprezentuje přijímající výpočet } M(x)$$

- Navíc platí, že mezi modely $\mathbf{a}$ formule φ a přijímajícími výpočty $M(x)$ je bijekce
- Platí tedy $f(x) = \#\mathrm{SAT}(\varphi)$
- Následující algoritmus vyčísluje $f(x)$:
  1. V polynomiálním čase zkonstruuj φ
  2. Vrať $\#\mathrm{SAT}(\varphi)$

$$f \leq_P \#\mathrm{SAT}$$

Parsimonious převod

Definice

Polynomiální převod z problému A na problém B je parsimonious, pokud zachovává počet certifikátů.

- Je-li A převoditelný na B parsimonious převodem, pak $\#A \leq_P \#B$
- Převod z $A \in \mathrm{NP}$ do SAT, který jsme popsali v důkazu Cookovy-Levinovy věty, je parsimonious
- funkce $\#\mathrm{SAT}$ je proto $\#\mathrm{P}$-úplná

Disjunktivní normální forma

Literál výroková proměnná x nebo její negace $\neg x$

Term konjunkce literálů

- Například $x \wedge \neg y \wedge \neg z$
- Prázdný term je splněný ($\top$)

DNF formule je v disjunktivní normální formě, pokud jde o disjunkci termů

Příklad

Následující formule je v DNF

$$\psi = x \vee (\neg y \wedge z) \vee (\neg x \wedge y) \vee (\neg x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg z)$$

Splnitelnost DNF

- Splnitelnost DNF ψ je možné ověřit v polynomiálním čase
  - Ověříme, zda je splnitelný jeden z termů ψ
  - Term T je splnitelný $\iff$ T neobsahuje $x \wedge \neg x$ pro žádnou proměnnou x
- Pro danou KNF φ lze jednoduše zkonstruovat DNF $\psi \equiv \neg\varphi$
  - Použijeme De Morganova pravidla

$$\neg(a \wedge b) \equiv \neg a \vee \neg b \quad \text{and} \quad \neg(a \vee b) \equiv \neg a \wedge \neg b$$

Uvažme KNF

$$\varphi = \neg x \wedge (y \vee \neg z) \wedge (x \vee \neg y) \wedge (x \vee y \vee z) \wedge (\neg x \vee z)$$

Aplikací De Morganových pravidel na $\neg\varphi$ dostaneme DNF $\psi \equiv \neg\varphi$

$$\psi = x \vee (\neg y \wedge z) \vee (\neg x \wedge y) \vee (\neg x \wedge \neg y \wedge \neg z) \vee (x \wedge \neg z)$$

Počítání modelů DNF

- Definujeme funkci $\#\,\text{DNF-SAT}$, která pro každou DNF ψ splňuje

$$\#\,\text{DNF-SAT}(\psi) = |\{\mathbf{a} \mid \psi(\mathbf{a}) = 1\}|$$

- $\#\,\text{DNF-SAT}$ patří do $\#\text{P}$

Věta

$\#\text{SAT} \leq_P \#\,\text{DNF-SAT}$, *tedy funkce* $\#\,\text{DNF-SAT}$ *je* $\#\text{P}$*-úplná.*

Výpočet $\#\text{SAT}(\varphi)$ s pomocí $\#\,\text{DNF-SAT}$

Vstup: KNF φ

1 $n \leftarrow$ počet proměnných v φ
2 $\psi \leftarrow$ DNF ekvivalentní $\neg\varphi$
3 **return** $2^n - \#\,\text{DNF-SAT}(\psi)$

Počet perfektních párování v bipartitním grafu

Perfektní párování v bipartitním grafu (BPM)

Instance: Bipartitní graf $G = (V = A \cup B, E \subseteq A \times B)$, kde $|A| = |B|$.

Otázka: Existuje v G párování velikosti $|A| = |B|$?

Věta (Bez důkazu)

Funkce $\#\,\mathrm{BPM}$ *je* $\#\mathrm{P}$*-úplná.*

Permanent matice

Definice

Je-li A matice typu $n \times n$ definujeme permanent A jako

$$\mathrm{perm}(A) = \sum_{\pi \in S(n)} \prod_{i=1}^{n} a_{i,\pi(i)},$$

kde $S(n)$ je množina permutací množiny $\{1, \ldots, n\}$.

- „Determinant“, kde neuvažujeme znaménko permutace.
- Je-li A matice sousednosti bipartitního grafu G, pak $\mathrm{perm}(A)$ určuje počet perfektních párování G.

Věta (Bez důkazu)

Funkce perm *je* $\#\mathrm{P}$*-úplná.*