1 Polynomiální hierarchie

1.1 Zavedení

Třída NP byla zavedena pomocí existenční kvantifikace na polynomiálně vyčíslitelný predikát. Třída coNP obsahuje doplňky jazyků z NP a lze tedy definovat pomocí obecné kvantifikace na polynomiálně vyčíslitelný predikát. Polynomiální hierarchie zobecňuje třídy NP a coNP tím, že budeme uvažovat několik existenčních a obecných kvantifikátorů na polynomiálně vyčíslitelný predikát.

Definice 1.1.1 Nechť L je jazyk nad abecedou A a $k \geq 0$.
(i) $L \in \Sigma_k^P$ právě tehdy, když existuje p-vyčíslitelná relace $R(w, x_1, \ldots, x_k)$ pro slova $w \in A^*$ a $x_i \in B^*$, kde B je libovolná konečná abeceda, a polynom $q(n)$ s celočíselnými koeficienty tak, že pro každé slovo $w \in A^*$ platí

$$w \in L \iff (\exists x_1 \in B^{q(|w|)})(\forall x_2 \in B^{q(|w|)}) \ldots (Q\, x_k \in B^{q(|w|)}) R(w, x_1, \ldots, x_k),$$

kde Q je $\forall$, pokud je k sudé a $\exists$, pokud je k liché.
(ii) $L \in \Pi_k^P$ právě tehdy, když $A^* \setminus L$ patří do Σ_k^P. Třída Π_k^P je tedy definována jako co-Σ_k^P.

Speciálně, $\Sigma_0^P = \Pi_0^P = \mathrm{P}$. Polynomiální hierarchií budeme rozumět posloupnost tříd Σ_k^P a Π_k^P a budeme ji značit PH.

Příklad jazyka, který patří do Σ_2^P, je jazyk slov, která kódují dvojice (G, k), kde G je graf a k je přirozené číslo a které splňují, že maximální klika v G má velikost k. Podmínka, která charakterizuje slova v tomto jazyce, je vyjádřitelná výrokem, že existuje množina vrcholů velikosti k, která je klikou v G, a každá množina vrcholů velikosti $k+1$ není klikou v G. Tento výrok lze zapsat ve formě, která odpovídá třídě Σ_2^P.

Příklad jazyka, který patří do Π_2^P, je jazyk slov, která kódují minimální Booleovské formule, tedy formule, pro které neexistuje ekvivalentní kratší formule. V tomto případě lze podmínku na formuli $\varphi(x_1, \ldots, x_n)$ vyjádřit tak, že pro každou kratší formuli $\psi(x_1, \ldots, x_n)$ existuje ohodnocení proměnných $x_1, \ldots, x_n$ tak, že $\varphi(x_1, \ldots, x_n) \neq \psi(x_1, \ldots, x_n)$.

Pro úplnost si ještě uveďme třídy Δ_k^P, které jsou pro $k \geq 1$ definované pomocí TS s orákulem následovně

$$\Delta_k^P = \mathrm{P}^{\Sigma_{k-1}^P} = \mathrm{P}^{\Pi_{k-1}^P}\ .$$

Speciálně, $\Delta_1^P = \mathrm{P}$ a formálně se definuje také $\Delta_0^P = \mathrm{P}$. Pro každé $k \geq 1$ platí

$$\Delta_k^P \subseteq \Sigma_k^P \cap \Pi_k^P\ ,$$

ale rovnost nemusí platit pro žádné $k \geq 1$. Speciálně, pro $k = 1$ dostáváme inkluzi $\mathrm{P} \subseteq \mathrm{NP} \cap \text{co-NP}$, která může být ostrá.

1.2 Vlastnosti

Platí následující inkluze.

Věta 1.2.1 *Pro každé $k \geq 0$ platí*

$$\Sigma_k^P \cup \Pi_k^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P \ .$$

Není známo, zda je polynomiální hierarchie nekonečná, tedy zda jsou všechny inkluze v této větě ostré. Nekonečnost PH lze vyjádřit jednodušeji jako hypotézu, že pro každé k je $\Sigma_k^P \neq \Sigma_{k+1}^P$. Pro $k = 0$ je tato hypotéza ekvivalentní P $\neq$ NP. Pro větší k se jedná o silnější tvrzení, protože pro každé $k \geq 0$ platí implikace

$$(\Sigma_{k+1}^P \neq \Sigma_{k+2}^P) \Rightarrow (\Sigma_k^P \neq \Sigma_{k+1}^P) \ .$$

Protože Π_k je co-Σ_k, lze nekonečnost PH ekvivalentně vyjádřit také tak, že pro každé k je $\Pi_k^P \neq \Pi_{k+1}^P$.

Každá ze tříd Σ_k^P a Π_k^P obsahuje úplné problémy vůči polynomiální převoditelnosti. Příkladem takové úlohy je test pravdivosti pro uzavřené kvantifikované Booleovské formule v prenexním tvaru, ve kterých lze posloupnost kvantifikátorů rozdělit do bloků stejných kvantifikátorů tak, že pořadí bloků existenčních a obecných kvantifikátorů je stejného typu jako kvantifikace definující uvažovanou třídu Σ_k^P nebo Π_k^P.

Věta 1.2.2 *Pro každé $k \geq 0$ je $\Sigma_k^P \cup \Pi_k^P \subseteq$ PSPACE.*

Důkaz. Pro libovolný jazyk $L \in \Sigma_k^P \cup \Pi_k^P$ lze $L \in$ PSPACE dokázat pomocí podobné rekurzivní funkce, jako v důkazu QBF $\in$ PSPACE. □

Věta 1.2.3 *Pokud je* PH *nekonečná, pak třída úloh, která je definována jako sjednocení $\bigcup_k \Sigma_k^P$, neobsahuje úplnou úlohu vůči p-převoditelnosti.*

Důkaz. Kdyby L byla úplná úloha pro třídu $\bigcup_k \Sigma_k^P$, pak $L \in \Sigma_k^P$ pro některé k. Pak každá úloha ze Σ_{k+1}^P je p-převoditelná na L, a tedy patří do Σ_k^P, tedy $\Sigma_k^P = \Sigma_{k+1}^P$. To je ve sporu s předpokladem nekonečnosti PH. □

Pokud je PH nekonečná, pak z této věty plyne, že $\bigcup_k \Sigma_k^P$ je vlastní podtřída PSPACE, protože PSPACE obsahuje úplné úlohy vůči p-převoditelnosti.

2 Složitost aproximačních úloh

2.1 Neaproximovatelnost Max-SAT v P

Připomeňme, že SAT je problém splnitelnosti pro formule v CNF a 3-SAT pro formule v CNF s právě třemi literály v každé klausuli. Pro optimalizační problém Max-SAT, tj. problém nalezení maximálního počtu klausulí, které lze současně splnit, používáme jinou konvenci. Označením Max-3-SAT se rozumí problém Max-SAT pro formule s klausulemi s nejvýše třemi literály, tj. připouštíme též klausule s jedním a dvěma literály. Pokud v kontextu Max-SAT je potřeba mluvit o formulích s právě třemi literály v každé klausuli, budeme používat označení Max-E3-SAT (exactly).

Pro libovolnou formuli φ v CNF rozumíme zápisem Max-SAT(φ) maximální počet současně splnitelných klausulí ve formuli φ a zápisem $|\varphi|$ označujeme počet klausulí formule φ.

Pro problém Max-SAT jsme si uváděli polynomiální aproximační algoritmus s faktorem 3/4. Pokud se omezíme pouze na klausule délky 3, lze aproximační faktor zlepšit.

Věta 2.1.1 *Pro problém Max-E3-SAT existuje polynomiální aproximační algoritmus s faktorem* $7/8$.

Důkaz. Pokud ohodnotíme proměnné náhodně a nezávisle tak, že pravděpodobnost jedničky je pro každou proměnnou $1/2$, pak pravděpodobnost splnění klausule délky k je $1 - 1/2^k$. Pro $k = 3$ tedy dostaneme pravděpodobnost splnění jedné klausule $7/8$. Střední hodnota počtu splněných klausulí v libovolné instanci Max-E3-SAT s m klausulemi je tedy $7/8 \cdot m$. Pomocí metody postupného upřesňování lze splnit alespoň $7/8 \cdot m$ klausulí pomocí deterministického algoritmu. □

Věta 2.1.2 *Pokud P≠NP, pak pro žádné* $\delta > 0$ *neexistuje polynomiální aproximační algoritmus pro Max-E3-SAT, a tedy ani pro Max-SAT, s faktorem* $7/8 + \delta$.

Tuto větu odvodíme jako důsledek následující věty.

Věta 2.1.3 *Pro každé* $\varepsilon > 0$ *existuje polynomiálně vyčíslitelná funkce* τ *tak, že pro každou instanci* φ *problému SAT je* $\tau(\varphi)$ *instance problému Max-E3-SAT a platí*

$$\begin{aligned} \varphi \in SAT &\Rightarrow \text{Max-SAT}(\tau(\varphi)) \geq (1-\varepsilon)|\tau(\varphi)| \\ \varphi \notin SAT &\Rightarrow \text{Max-SAT}(\tau(\varphi)) \leq (7/8+\varepsilon)|\tau(\varphi)|. \end{aligned}$$

Ukážeme nejprve, že z Věty 2.1.3 plyne Věta 2.1.2 jako jednoduchý důsledek. *Důkaz Věty 2.1.2 na základě Věty 2.1.3.* Důkaz sporem. Předpokládejme, že existuje aproximační algoritmus A pro Max-E3-SAT s faktorem $7/8 + \delta$. Pro libovolnou instanci Max-E3-SAT τ označme jako $A(\tau)$ počet klausulí, které splní ohodnocení nalezené algoritmem A. Zvolme $\varepsilon = \delta/(2 + \delta)$. Lze ověřit, že pak platí

$$\left(\frac{7}{8} + \delta\right)(1 - \varepsilon) = \frac{7}{8} + \frac{9}{8} \cdot \frac{\delta}{2 + \delta} = \frac{7}{8} + \frac{9}{8} \cdot \varepsilon > \frac{7}{8} + \varepsilon \ . \tag{1}$$

Nechť τ je funkce splňující podmínky Věty 2.1.3 pro zvolené ε. Pro libovolnou instanci φ problému SAT dokážeme

$$\varphi \in SAT \iff \mathrm{A}(\tau(\varphi)) > (7/8 + \varepsilon)|\tau(\varphi)| \ . \tag{2}$$

Pokud $\varphi \in SAT$, pak z volby funkce τ a z Věty 2.1.3 plyne, že maximální počet splnitelných klausulí v $\tau(\varphi)$ je alespoň $(1 - \varepsilon)|\tau(\varphi)|$. Protože A má aproximační faktor $7/8 + \delta$, plyne z (1) nerovnost $\mathrm{A}(\tau(\varphi)) > (7/8 + \varepsilon)|\tau(\varphi)|$.

Pokud $\varphi \notin SAT$, pak z Věty 2.1.3 plyne $\text{Max-SAT}(\tau(\varphi)) \leq (7/8 + \varepsilon)|\tau(\varphi)|$. Protože A nemůže splnit více klausulí než optimální řešení, platí také $\mathrm{A}(\tau(\varphi)) \leq (7/8 + \varepsilon)|\tau(\varphi)|$.

Ekvivalence (2) umožňuje testovat SAT v polynomiálním čase, což je spor s předpokladem P≠NP. Za předpokladu P≠NP tedy algoritmus A s uvedenými vlastnostmi nemůže neexistovat. □

Větu 2.1.3 odvodíme z analogické Věty 2.1.6 pro soustavy lineárních rovnic nad dvouprvkovým tělesem $Z_2 = \{0, 1\}$, jehož sčítání je definováno jako sčítání modulo 2 a v někdy se též označuje jako XOR (exclusive OR). Pro libovolnou soustavu lineárních rovnic L nad Z_2 označme jako $|L|$ počet rovnic L a jako $\text{Max-Lin}(L)$ maximální počet rovnic, které lze splnit některým ohodnocením proměnných. Obecně připouštíme, že některá rovnice se může v soustavě vyskytovat vícekrát. Pokud je taková rovnice splněna, počítá se do $\text{Max-Lin}(L)$ každý její výskyt zvlášť.

Uveďme si příklad soustavy čtyř lineárních rovnic s proměnnými x_i pro $i = 1, \ldots, 6$, kterou zapíšeme pomocí matice soustavy a vektoru pravých stran a budeme ji označovat L_1.

$$\left(\begin{array}{cccccc|c} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right) \tag{3}$$

Gaussovou eliminací můžeme soustavu L_1 upravit do tvaru

$$\left(\begin{array}{cccccc|c} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \tag{4}$$

Upravená soustava nemá řešení, protože poslední rovnice má nulovou levou stranu a nenulovou pravou stranu. Protože původní a upravená soustava jsou ekvivalentní, nemá řešení ani soustava původní. Platí tedy Max-Lin$(L_1) \leq 3$. Na druhé straně, ohodnocení proměnných $x_5 = 1$ a $x_i = 0$ pro $i \neq 5$ splní první tři rovnice původní soustavy. Platí tedy Max-Lin$(L_1) = 3$.

Označme jako Max-Lin-2 úlohu zjistit Max-Lin(L) pro libovolnou danou soustavu L nad Z_2 a jako Max-Ek-Lin-2 stejnou úlohu, ale jen pro soustavy, jejichž každá rovnice závisí právě na k proměnných. Budeme se zabývat výhradně případem $k = 3$. Například výše uvedená soustava L_1 je instancí Max-E3-Lin-2.

Hlavní tvrzení týkající se problému Max-E3-Lin-2, které uvedeme, je Věta 2.1.6, pomocí které dokážeme Větu 2.1.3. Nejprve ale dokážeme některé jednodušší vlastnosti úloh Max-Lin-2 a Max-E3-Lin-2.

Věta 2.1.4 *Úloha Max-Lin-2 je NP-úplná.*

Důkaz. Ukážeme převod libovolné instance 3-SAT na Max-Lin-2, tak, že každou klausuli nahradíme množinou 7 lineárních rovnic. Konkrétně, klausuli $l_1 \vee l_2 \vee l_3$ nahradíme rovnicemi

$$\begin{array}{ccccccccc} l_1 & & & & & & & = & 1 \\ & & l_2 & & & & & = & 1 \\ & & & & & & l_3 & = & 1 \\ l_1 & \oplus & l_2 & & & & & = & 1 \\ l_1 & \oplus & & & & & l_3 & = & 1 \\ & & l_2 & \oplus & & & l_3 & = & 1 \\ l_1 & \oplus & l_2 & \oplus & & & l_3 & = & 1 \end{array}$$

Lze ověřit, že pokud není splněn žádný z literálů l_1, l_2, l_3, není splněna žádná z uvedených rovnic. Na druhé straně, každé ohodnocení, které splní alespoň jeden z literálů, splní právě 4 z těchto rovnic. Instance 3-SAT obsahující m klausulí, je tímto způsobem převedena na soustavu $7m$ rovnic a platí, že vstupní instance 3-SAT je splnitelná právě tehdy, pokud je možné splnit alespoň (a tedy právě) $4m$ z uvedených rovnic. □

Věta 2.1.5 *Pro úlohu Max-E3-Lin-2 existuje polynomiální aproximační algoritmus s faktorem* $1/2$.

Důkaz. Pokud ohodnotíme proměnné náhodně a nezávisle tak, že pravděpodobnost jedničky je pro každou proměnnou $1/2$, pak je pravděpodobnost splnění každé rovnice v instanci Max-E3-Lin-2 rovna $1/2$. Pokud má vstupní soustava m rovnic, je střední hodnota počtu splněných rovnic $m/2$. Pomocí metody postupného upřesňování lze splnit $m/2$ rovnic pomocí deterministického algoritmu. Pro tento účel je ale potřeba rovnice nad tělesem Z_2 převést na rovnice nad reálnými

čísly tvořené z multilineárních polynomů. To je možné, protože pro libovolné $x_1, x_2, x_3 \in \{0,1\}$ platí

$$x_1 \oplus x_2 \oplus x_3 = \frac{1-(1-2x_1)(1-2x_2)(1-2x_3)}{2} .$$

Polynom v pravé straně tedy vyjadřuje splnění rovnice $x_1 \oplus x_2 \oplus x_3 = 1$. Splnění rovnice $x_1 \oplus x_2 \oplus x_3 = 0$ je pak vyjádřeno polynomem

$$1-(x_1 \oplus x_2 \oplus x_3) = \frac{1+(1-2x_1)(1-2x_2)(1-2x_3)}{2} .$$

Pokud sečteme polynomy, které vyjadřují splnění jednotlivých rovnic v soustavě, dostaneme multilineární polynom se střední hodnotou $m/2$ a na něj použijeme metodu postupného upřesňování popsanou společně s aproximačními algoritmy pro Max-SAT. □

Věta 2.1.6 *Pro každé $\varepsilon > 0$ existuje polynomiálně vyčíslitelná funkce L tak, že pro každou instanci φ problému SAT je $L(\varphi)$ instance problému Max-E3-Lin-2 a platí*

$$\begin{aligned} \varphi \in SAT &\Rightarrow \text{Max-Lin}(L(\varphi)) \geq (1-\varepsilon)|L(\varphi)| \\ \varphi \notin SAT &\Rightarrow \text{Max-Lin}(L(\varphi)) \leq (1/2+\varepsilon)|L(\varphi)|. \end{aligned}$$

Důkaz této věty nebudeme uvádět, ale použijeme jí k důkazu Věty 2.1.3.

Důkaz Věty 2.1.3 na základě Věty 2.1.6. Nechť φ je vstupní instance 3-SAT a $L(\varphi)$ odpovídající instance Max-E3-Lin-2, která má vlastnosti popsané ve Větě 2.1.6. Převedeme $L(\varphi)$ na instanci Max-E3-SAT $\tau(\varphi)$ tak, že každou rovnici nahradíme pomocí CNF složené ze 4 klausulí, která vyjadřuje splnění dané rovnice. Konkrétně, rovnici

$$x_1 \oplus x_2 \oplus x_3 = 1$$

nahradíme formulí

$$(x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3)(x_1 \vee \overline{x}_2 \vee \overline{x}_3)$$

a rovnici

$$x_1 \oplus x_2 \oplus x_3 = 0$$

nahradíme formulí

$$(\overline{x}_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee x_3)(x_1 \vee x_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3) .$$

Pokud je φ splnitelná, pak podle Věty 2.1.6 lze splnit alespoň $(1-\varepsilon)|L(\varphi)|$ rovnic. Každá splněná rovnice reprezentuje 4 splněné klausule, tedy platí

$$\text{Max-SAT}(\tau(\varphi)) \geq 4(1-\varepsilon)|L(\varphi)| = (1-\varepsilon)|\tau(\varphi)| .$$

Pokud není φ splnitelná, pak podle Věty 2.1.6 lze splnit nejvýše $(1/2+\varepsilon)|L(\varphi)|$ rovnic a tedy pro každé ohodnocení je alespoň $(1/2-\varepsilon)|L(\varphi)|$ rovnic nesplněno. Každá nesplněná rovnice reprezentuje alespoň jednu nesplněnou klausuli, tedy platí

$$\text{Max-SAT}(\tau(\varphi)) \leq |\tau(\varphi)| - \left(\frac{1}{2} - \varepsilon\right)|L(\varphi)| = \left(1 - \frac{\frac{1}{2} - \varepsilon}{4}\right)|\tau(\varphi)|$$

a tedy také

$$\text{Max-SAT}(\tau(\varphi)) \leq \left(\frac{7}{8} + \frac{\varepsilon}{4}\right)|\tau(\varphi)| \leq \left(\frac{7}{8} + \varepsilon\right)|\tau(\varphi)| \ .$$

Tím je dokázáno, že formule $\tau(\varphi)$ má vlastnosti požadované ve Větě 2.1.3. □

2.2 Neaproximovatelnost problému kliky v P

Věta 2.2.1 *Pokud P≠NP, pak pro žádné $\delta > 0$ neexistuje polynomiální aproximační algoritmus pro problém kliky s faktorem $7/8 + \delta$.*

Důkaz. Použijeme redukci SAT na problém kliky, která vytváří graf, jehož uzly jsou literály a hrany jsou mezi literály, které se nevylučují a jsou v různých klausulích. Pokud tuto redukci použijeme pro Max-SAT, pak tato redukce přenáší počet splněných klausulí na velikost kliky. Pokud by tedy existoval polynomiální aproximační algoritmus pro problém kliky s faktorem $7/8+\delta$, existoval by takový algoritmus i pro Max-SAT. Podle Věty 2.1.2 takový algoritmus za předpokladu P≠NP neexistuje. □

Pro zesílení tohoto výsledku definujme součin grafů následujícího typu.

Definice 2.2.2 Nechť pro $i = 1, \ldots, k$ jsou $G_i = (V_i, E_i)$ grafy. Pak definujeme graf $G = G_1 \boxtimes \ldots \boxtimes G_k$ na množině vrcholů $V = V_1 \times \ldots \times V_k$ tak, že vrcholy $(u_1, \ldots, u_k)$ a $(v_1, \ldots, v_k)$ tvoří v G hranu právě tehdy, když pro každé $i = 1, \ldots, k$ je buď $u_i = v_i$ nebo jsou vrcholy u_i a v_i spojeny v G_i hranou.

Značení tohoto součinu pomocí symbolu $\boxtimes$ odpovídá tvaru grafu, který vznikne součinem dvou grafů, které obsahují jedinou hranu. Velikost maximální kliky v libovolném grafu G budeme označovat $k_{\max}(G)$.

Věta 2.2.3 *Pro libovolné grafy $G_1, \ldots, G_k$ platí*

$$k_{\max}(G_1 \boxtimes \ldots \boxtimes G_k) = k_{\max}(G_1) \cdot \ldots \cdot k_{\max}(G_k).$$

Důkaz. Pro $i = 1, \ldots, k$ nechť A_i je některá maximální klika v G_i. Pak $A_1 \times \ldots \times A_k$ je klikou v $G_1 \boxtimes \ldots \boxtimes G_k$, což dokazuje

$$k_{\max}(G_1 \boxtimes \ldots \boxtimes G_k) \geq k_{\max}(G_1) \cdot \ldots \cdot k_{\max}(G_k).$$

Pro opačný směr označme jako A libovolnou maximální kliku v $G_1 \boxtimes \ldots \boxtimes G_k$ a pro $i = 1, \ldots, k$ nechť A_i je projekce A do i-té souřadnice. Pak platí $A \subseteq A_1 \times \ldots \times A_k$. Navíc, každá z množin A_i je klikou v příslušném G_i. Z toho plyne požadovaná nerovnost $\leq$. □

Důsledek 2.2.4 *Pokud P≠NP, pak pro žádnou konstantu $\varepsilon > 0$ neexistuje polynomiální aproximační algoritmus pro problém kliky s faktorem ε.*

Důkaz. Důkaz sporem. Předpokládejme, že existuje algoritmus A s faktorem ε. Pro libovolný graf G označme jako $A(G)$ velikost kliky, kterou algoritmus A v G nalezne. Ukážeme, že pak existuje algoritmus s faktorem $7/8 + \delta$ pro nějaké kladné δ, což bude spor s Větou 2.2.1.

Pro libovolné přirozené číslo t plyne z Věty 2.2.3 a z předpokladu, že A je aproximační algoritmus s faktorem ε, následující nerovnost

$$\varepsilon \cdot k_{\max}(G)^t \leq \mathrm{A}(G^t) \leq k_{\max}(G)^t,$$

kde G^t je $\boxtimes$ součin t kopií grafu G. Platí tedy také

$$\varepsilon^{1/t} \cdot k_{\max}(G) \leq \mathrm{A}(G^t)^{1/t} \leq k_{\max}(G).$$

Pro každé $\varepsilon > 0$ existuje t tak, že $\varepsilon^{1/t} > 7/8$. Pro takovou volbu t dává hodnota $\mathrm{A}(G^t)^{1/t}$ aproximaci veličiny $k_{\max}(G)$ s faktorem $7/8+\delta$. Aproximační algoritmus, který nalezne konkrétní kliku v G, dostaneme tak, že vezmeme kliku, kterou najde A použitý na graf G^t a vybereme největší projekci této kliky do jedné souřadnice. Tato projekce je klika v G velikosti alespoň $\mathrm{A}(G^t)^{1/t}$. Protože t je konstanta, je tento algoritmus také polynomiální, stejně jako předpokládaný algoritmus A. □

2.3 Zmínka o PCP větě

Výše uvedená Věta 2.1.6 je důsledkem vět, které se týkají třídy PCP (Probabilistically Checkable Proofs). Tato třída je, podobně jako NP, založena na verifikaci důkazů příslušnosti do jazyka. Verifikátor pro NP popsaný v jedné z předchozích kapitol dostává důkaz x tvrzení $w \in L$ polynomiální délky, má možnost jej přečíst celý a v polynomiálním čase musí rozhodnout, zda jde o korektní důkaz tvrzení $w \in L$ nebo ne. Verifikátor pro třídu PCP může dostat důkaz x tvrzení $w \in L$ i exponenciální délky, ale může z něj přečíst jen omezený počet symbolů. Důkaz x je zapsán v abecedě $\{0, 1\}$. Verifikátor pracuje pravděpodobnostně, tj.

dostane také řetěz u náhodných bitů omezené délky a požadavek na jeho funkci je formulován tak, že chybný důkaz x musí rozpoznat s určitou předem danou pravděpodobností. Vzhledem k tomu, že verifikátor nemá možnost přečíst důkaz celý, tak spolehlivé rozpoznání správnosti důkazu ani není možné. Verifikátor je deterministický TS, který má tři vstupní pásky (pro w, pro důkaz x a pro náhodné bity u). Pravděpodobnostní chování se dosahuje pouze tím, že vstup u je náhodný. Pásky pro w a u čte obvyklým způsobem. Pásku pro x čte tak, že na pracovní pásku zapíše indexy několika symbolů v binární soustavě a v jednom kroku získá hodnoty všech těchto symbolů z pásky x. Přístup k této pásce je tedy podobný jako přístup RAM do jeho registrů, až na to, že musí všechny požadované symboly přečíst současně a další symboly z pásky x již nemůže získat. Bez újmy na obecnosti můžeme požadovat, že současně se zapsáním indexů požadovaných symbolů z x zapíše na pásku také tabulku funkce $\{0,1\}^s \to \{0,1\}$ podle které bude ze získaných symbolů odvozovat výstupní rozhodnutí. Výstup 1 znamená přijetí důkazu, výstup 0 jeho zamítnutí.

Definice 2.3.1 Nechť $r(n), s(n)$ jsou nezáporné funkce. Pak třídou $\mathrm{PCP}(r(n), s(n))$ rozumíme třídu jazyků L, pro které existuje verifikátor R s vlastnostmi popsanými výše a který navíc splňuje:

1. Délka vstupní posloupnosti náhodných bitů u je $r(|w|)$.
2. Při libovolném výpočtu pro slovo w přečte $s(|w|)$ symbolů důkazu x v abecedě $\{0,1\}$.
3. Pokud $w \in L$, pak existuje důkaz x tak, že $\Pr_u(R(w,x,u)=1)=1$ (důkaz je přijat vždy).
4. Pokud $w \notin L$, pak pro každý důkaz x platí $\Pr_u(R(w,x,u)=1) < 1/2$ (chyba v důkazu je nalezena s pstí větší než $1/2$).

Tuto definici využijeme tak, že budeme připouštět více různých funkcí $r(n)$ a $s(n)$, ale budeme uvažovat jen funkce velmi omezené velikosti. Konkrétně, budeme připouštět libovolnou $r(n) = O(\log n)$ a libovolnou $s(n) = O(1)$. Takto získanou třídu budeme značit $\mathrm{PCP}(O(\log n), O(1))$. Následující věta je základní forma věty o PCP, která byla použita k důkazu prvních výsledků o neaproximovatelnosti Max-SAT, které jsou slabší než Věta 2.1.2.

Věta 2.3.2 (PCP) *$NP = PCP(O(\log n), O(1))$.*

Věta 2.1.6, pomocí které se dokazuje Věta 2.1.2, plyne z věty, která je analogická výše uvedené větě o PCP, ale obsahuje další parametry.

3 Vzájemné simulace jedno a vícepáskového TS a RAM

K porovnání výpočetní síly modelů se používají vzájemné simulace. Pokud může být libovolný výpočet v jednom modelu simulován výpočtem v jiném modelu bez podstatného nárůstu složitosti, znamená to, že druhý model je alespoň tak silný jako model první. V této kapitole dokážeme výsledky o simulaci vícepáskového TS pomocí TS s jednou nebo dvěma páskami a dále simulaci RAM pomocí TS a naopak, které jsou v textu Výpočtová složitost I pouze formulovány.

3.1 Programovací jazyk pro TS

Pro zápis programu jednopáskového TS, použijeme jednoduchý program bez rekurze, který může používat konečný počet proměnných, které mohou uchovávat znaky nějaké konečné abecedy. Pro čtení z pásky a zápis na pásku použijeme speciální proměnnou, například H, která bude reprezentovat čtený symbol na pásce. Hodnota H bude vždy symbol na aktuálně čtené pozici a dosazení do H bude znamenat zápis znaku na pásku. Pro pohyb čtecí hlavy budeme používat funkce *vlevo* a *vpravo*.

Následující tvrzení je zjednodušenou verzí tvrzení, které použijeme v další sekci, v tom, že budeme používat pouze jednu proměnnou pro čtení a zápis symbolů na pásku. TS, který budeme popisovat v další sekci, bude mít pásku rozdělenou na stopy a bude proto mít více proměnných označujících symboly na jednotlivých stopách. Pro vysvětlení důkazu je uvedené zjednodušení postačující. Rozšíření se týká jen technických detailů.

Lemma 3.1.1 *Každý program bez rekurze, který má konečný počet proměnných s konečným oborem hodnot a který ovládá čtecí hlavu na pásce výše popsaným způsobem, je ekvivalentní vhodnému TS.*

Důkaz. Program převedeme na vývojový diagram a vytvoříme abstraktní zařízení, které se skládá ze tří částí: vývojový diagram, blok proměnných a páska. Každá z těchto částí se může nacházet v různých stavech. Stav vývojového diagramu je určen uzlem, který je právě prováděn. Množinu všech těchto uzlů označme U. Stav bloku proměnných je určen vektorem hodnot všech proměnných. Množinu všech kombinací hodnot všech proměnných označme D. Protože obor hodnot všech proměnných je konečný, je také D konečná. Stav pásky je určen jejím obsahem a polohou hlavy.

Popsané zařízení lze rozdělit na řídící jednotku a datové struktury dvěma způsoby. První způsob je, že proměnné a pásky považujeme za datové struktury, se kterými pracuje vývojový diagram pomocí povolených operací. Tento pohled odpovídá tomu, jak je běžně chápán vývojový diagram, a je užitečný při návrhu celého zařízení a pro porozumění jeho činnosti.

Druhý způsob pohledu je, že vývojový diagram a blok proměnných považujeme za řídící jednotku, která se nachází v některém z možných stavů z množiny $Q = U \times D$ a která svůj stav a stav pásek mění podle přechodové funkce

$$f : Q \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\} .$$

Stav takto chápané řídící jednotky zaznamenává uzel vývojového diagramu, ve kterém se výpočet nachází, a okamžité hodnoty všech proměnných. Protože tato řídící jednotka je konečná, může být použita jako řídící jednotka TS. Přechodovou funkci tohoto TS získáme tak, že uvážíme všechny možné kombinace hodnot $U \times D$ a čteného symbolu na pásce, který je reprezentován proměnnou H a vyjádříme odpovídající akci, kterou popsané zařízení provede.

V dalším budeme předpokládat, že proměnná je pouze jedna, je nazvána X, její obor hodnot je $\mathcal{X}$ a je tedy $Q = U \times \mathcal{X}$. Uzly vývojového diagramu mohou být například následujících typů.

1. test, zda znak na pozici hlavy je v dané množině znaků
2. test, zda znak v proměnné X je v dané množině znaků
3. krok vlevo nebo vpravo
4. zápis znaku na pozici hlavy
5. zápis znaku do proměnné X
6. koncový uzel

Program TS bude popsán přechodovou funkcí, tedy funkcí

$$f : Q \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\} .$$

Popišme, jak budou v tabulce přechodové funkce realizovány akce odpovídající uzlu diagramu u, pokud $X = x$ a čtený symbol je a. Výchozí stav TS je dvojice tvaru (u, x). Popis akce je odlišný pro různé typy uzlu u. Popisy pro jednotlivé typy uzlů jsou v následujícím seznamu. Pokud je u typu test, je u_0 jeho následník pro nesplnění podmínky a u_1 následník pro splnění podmínky. Následník ostatních typů uzlů bude označen jako u_1.

1. test, zda znak na pozici hlavy je v dané množině znaků
   $f((u, x), a)$ je $((u_0, x), a, 0)$, pokud a není v testované množině a $((u_1, x), a, 0)$ pokud a je v testované množině.
2. test, zda znak v proměnné X je v dané množině znaků
   $f((u, x), a)$ je $((u_0, x), a, 0)$, pokud x není v testované množině a $((u_1, x), a, 0)$ pokud x je v testované množině.

3. krok vlevo nebo vpravo
   $f((u, x), a)$ je $((u_1, x), a, -1)$ nebo $((u_1, x), a, 1)$ podle směru přesunu hlavy, který je požadován ve u.

4. zápis znaku b na pozici hlavy
   $f((u, x), a)$ je $((u_1, x), b, 0)$, kde b je znak zapisovaný na pásku u uzlu u.

5. zápis znaku y do proměnné X
   $f((u, x), a)$ je $((u_1, y), a, 0)$, kde y je znak zapisovaný do X v uzlu u.

6. koncový uzel u
   $f((u, x), a)$ je nedefinováno.

Pokud opět uvážíme obecný případ s více proměnnými, je množina stavů získaného TS obecně exponenciálně veliká ve vztahu k počtu proměnných v programu. Změna hodnoty kterékoli proměnné nebo větvení vývojového diagramu podle hodnoty některé proměnné jsou realizovány jako změna vnitřního stavu řídící jednotky. Vzhledem k tomu, že změna jedné proměnné nebo větvení podle jedné proměnné může proběhnout při libovolné kombinaci hodnot ostatních proměnných, musí být přechodová funkce definována odpovídajícm způsobem pro všechny jejich kombinace. Pokud šlo o změnu hodnoty jedné proměnné, přejde výpočet do stavu, ve kterém je patřičně změněna příslušná proměnná, ostatní proměnné jsou zachovány a uzel diagramu je nastaven na následující uzel diagramu. Pokud šlo o větvení, jsou zachovány hodnoty všech proměnných, ale uzel diagramu, který je zaznamenán ve stavu, je nastaven na ten uzel, kam přejde vývojový diagram v závislosti na testované proměnné.

Podle definice TS čte každá instrukce symbol na pásce a nějaký symbol na pásku zapíše. Ve výše popsané konstrukci připouštíme, že některý krok výpočtu popsaného zařízení neprovede žádnou akci s páskou stroje. V přechodové funkci výsledného TS odpovídají takovéto kroky akcím, které formálně přečtou symbol z pásky, tentýž symbol opět na pásku zapíší a neprovedou žádný posun hlav. Protože je čtený symbol argumentem přechodové funkce, musí být v této funkci příslušná akce reprezentována opět pro všechny jeho možné hodnoty.

Popsaným způsobem získáme řídící jednotku TS s velkým, ale konečným počtem stavů a instrukcí. □

3.2 Simulace vícepáskového TS pomocí TS s jednou nebo dvěma páskami

Platí následující tvrzení.

Věta 3.2.1 *Každý vícepáskový TS M lze simulovat*
1. TS M' s jednou páskou

2. TS M'' se dvěma pracovními páskami v abecedě $\{0,1\}$*,*
tak, že výpočet o t krocích je simulován
1. výpočtem M' délky $O(t^2)$ kroků,
2. výpočtem M'' délky $O(t \log t)$ kroků.

Důkaz. Dokazovat budeme pouze tvrzení 1. Nechť stroj M má k pásek. Požadovaný stroj M' vytvoříme jako stroj, který bude mít jednu pásku s $2k$ stopami. Vždy dvě po sobě jdoucí stopy odpovídají jedné pásce stroje M. První z těchto stop je kopií obsahu příslušné pásky a druhá obsahuje symbol 1 na pozici, kde má M hlavu na této pásce a jinde obsahuje ε, tj. znak označující prázdné pole pásky.

Popíšeme simulaci jednoho kroku stroje M strojem M'. Díky analýze z předchozí sekce můžeme řídící jednotku stroje M' popsat programem s následujícími proměnnými. Proměnné $A_1, \ldots, A_k$ budou reprezentovat symboly, které stroj M' čte a zapisuje na těch stopách pásky, které odpovídají páskám $1, \ldots, k$ stroje M. Proměnné $H_1, \ldots, H_k$ budou reprezentovat symboly na stopách, které určují polohy jednotlivých hlav M, tedy stopy se symboly ε a 1. Program bude dále využívat proměnné $a_1, \ldots, a_k$ pro dlouhodobější ukládání symbolů z pásek stroje M, proměnné $u_1, \ldots, u_k$ pro pomocné hodnoty 0, 1 a proměnnou q pro stav jednotky stroje M.

Simulace jednoho kroku stroje M se skládá ze dvou fází. V první fázi stroj projde pásku zleva doprava až na všech stopách nalezne symbol 1, který označuje hlavu na příslušné stopě. Přitom postupně ukládá do svých proměnných symboly, které stroj M čte na jednotlivých páskách.

Po skončení tohoto úseku programu jsou v proměnných $a_1, \ldots, a_k$ zapsány symboly, které čte stroj M. Na základě těchto symbolů a stavu q vyhledá M' odpovídající instrukci M podle jeho přechodové funkce. Pro tento účel bude program testovat proměnné $q, a_1, \ldots, a_k$ a pro každou kombinaci jejich hodnot provede jiný blok příkazů, ve kterém hodnoty proměnných $a_1, \ldots, a_k$ změní na hodnoty, které mají být na jednotlivé stopy zapsány, a do proměných $s_1, \ldots, s_k$ zapíše posuny, které mají být na jednotlivých stopách uskutečněny. Pak M' zahájí cyklus, v jehož každém kroku posune hlavu o jednu pozici vlevo, přičemž na pozicích, kde některá z proměnných H_i má hodnotu 1, provádí navíc naplánované změny konfigurace simulovaného stroje M. Tyto změny si mohou vyžádat změny i na některém ze sousedních polí, což se realizuje pohybem na toto pole, příslušnými změnami a návratem zpět. Změny na více stopách na téže pozici pásky se provádí postupně. Cyklus končí, když jsou potřebné úpravy provedeny na všech stopách pásky. Pokud stroj M neučiní krok vlevo na nejlevější buňce žádné pásky, neučiní takový krok ani M'.

Tento cyklus dokončuje simulaci jednoho kroku stroje M. Celý postup simulace jednoho kroku stroje M je zapsán programem na Obrázku 1. Opakováním simulace jednoho kroku v dalším cyklu pak dostaneme simulaci celého výpočtu stroje M. □

```
u_0 ← 0; ... u_k ← 0
repeat
    if (H_1 = 1) { a_1 ← A_1; u_1 ← 1 }
    ...
    if (H_k = 1) { a_k ← A_k; u_k ← 1 }
    vpravo
until (u_1 = 1 and ... and u_k = 1)
(a_1, ..., a_k, s_1, ..., s_k) ← tabulka[q, a_1, ..., a_k] // dosazení do několika proměnných
repeat
    vlevo
    for i = 1 to k do {
        if (H_i = 1 and u_i = 1) {
            A_i ← a_i
            u_i ← 0
            if (s_i ≠ 0) H_i ← 0
            if (s_i = 1) {
                vpravo
                H_i ← 1
                vlevo
            }
            if (s_i = −1) {
                vlevo
                H_i ← 1
                vpravo
            }
        }
    }
until (u_1 = 0 and ... and u_k = 0)
```

Obrázek 1: Simulace jednoho kroku M.

3.3 Simulace TS pomocí RAM a naopak

Věta 3.3.1 *Jestliže nějaká úloha je řešitelná na TS v čase t, pak je řešitelná na RAM s jednotkovou mírou v čase $O(t)$. Simulující stroj navíc splňuje podmínku na polynomiální omezení velikosti čísel.*

Důkaz. Toto tvrzení je dokázáno v textu Výpočtová složitost I. □

Nyní zformulujeme a dokážeme obecnou simulaci RAM pomocí TS. Nejprve nebudeme předpokládat žádné omezení velikosti čísel a použijeme tedy bitovou (logaritmickou) míru složitosti.

Věta 3.3.2 *Jestliže nějaká úloha je řešitelná na RAM s bitovou mírou složitosti b, pak je řešitelná na vícepáskovém TS v čase $O(b^2)$.*

Důkaz. Simulující TS má 6 pásek. První páska obsahuje vstup zapsaný jako posloupnost binárně zapsaných čísel oddělených speciálními znaky. Na druhé pásce je zakódován obsah všech dosud použitých registrů stroje RAM. Tento kód se skládá z posloupnosti dvojic $\langle i, r_i \rangle$ pro všechny adresy i, pro které byl i-tý registr použit. Čísla ve dvojici jsou zapsána ve dvojkové soustavě. Za poslední dvojicí je znak označující konec pásky.

Třetí páska slouží k zápisu adresy při hledání hodnoty registru r_i nebo v_i. Čtvrtá, pátá a šestá páska slouží k ukládání čísel pro aritmetické operace. Např. při sčítání jsou operandy na čtvrté a páté pásce a součet se zapisuje na šestou. Tyto pásky budeme nazývat aritmetické.

Pro každou instrukci programu RAM obsahuje uvažovaný TS podprogram, který danou instrukci simuluje. Po ukončení každého podprogramu přejde TS do stavu, v němž začíná další z těchto podprogramů podle pořadí, jak jsou prováděny v RAM. Jestliže instrukce potřebuje obsah i-tého registru, je třeba prohledat druhou pásku a zjistit, zda je zde zapsána hodnota r_i. Při tomto hledání je číslo i zapsáno binárně na třetí pásce. Kdykoli je na druhé pásce nalezena nějaká dvojice $\langle i', r_{i'} \rangle$, je číslo i porovnáno s i'. Pokud nenastane rovnost, je druhá páska prohledávána dále. V opačném případě je hledané číslo r_i nalezeno. Pokud šlo o přímé adresování, je nalezené číslo r_i přeneseno na některou aritmetickou pásku k dalším operacím. Pokud šlo o nepřímé adresování, je r_i přeneseno na třetí pásku a je použito jako adresa pro druhé hledání. Pokud není r_i nalezeno vůbec, předpokládá se místo něj hodnota 0 a ta je zapsána na aritmetickou pásku.

Výsledek operace se zapisuje zpět do registrů. K zápisu hodnoty registru i na druhou pásku je třeba nejprve najít a přepsat mezerami případnou dříve zapsanou dvojici $\langle i, r_i \rangle$. Pak se nalezne znak konce pásky, vymaže se a za něj se zapíše dvojice $\langle i, r_i \rangle$ s aktuální hodnotou registru i a zápis se ukončí značkou konce pásky. Takto vznikají na pásce úseky popsané jen mezerami. Pro jednoduchost simulace ponecháme tyto úseky nevyužité.

Předpokládejme, že uvažovaný výpočet proběhl v k krocích, jejichž bitové složitosti byly b_i pro $i = 1, \ldots, k$. Platí tedy $b = \sum_{i=1}^{k} b_i$. Označme maximální délku obsahu druhé pásky s. Délka každého čísla, které je na druhé pásce zapsáno, byla započítána do složitosti některé instrukce. Pomocné symboly, tj. závorky, čárky a mezery prodlouží obsah druhé pásky nejvýše o konstantní násobek, proto $s = O(\sum_{i=1}^{k} b_i)$. Při simulaci jedné instrukce RAM je druhá páska prohledávána nejvýše dvakrát (kvůli nepřímému adresování) pro každý operand. Čas potřebný pro toto hledání a případné přenosy dat je $O(s)$. Simulaci jedné instrukce RAM na TS zahrnuje také výpočty na aritmetických páskách. Protože je délka čísel, se kterými pracuje i-tá instrukce, nejvýše rovna b_i, je počet kroků na příslušnou aritmetickou operaci nejvýše $O(b_i^2)$. Z toho již lze odvodit odhad na počet kroků TS ve tvaru $t = O(\sum_{i=1}^{k} b_i^2 + k \sum_{i=1}^{k} b_i)$. Protože $\sum_{i=1}^{k} b_i^2 \leq (\sum_{i=1}^{k} b_i)^2$ a $k \sum_{i=1}^{k} b_i \leq (\sum_{i=1}^{k} b_i)^2$ platí $t = O(b^2)$. □

Pokud RAM splňuje polynomiální omezení velikosti čísel, můžeme použít jednotkovou míru složitosti a platí následující tvrzení.

Věta 3.3.3 *Jestliže nějaká úloha je řešitelná na RAM s jednotkovou mírou složitosti k a jestliže RAM splňuje polynomiální omezení velikosti čísel, pak je tato úloha řešitelná na vícepáskovém TS v čase $O(k^2 \log k)$.*

Důkaz. Použijeme stejnou konstrukci jako v důkazu předchozí věty. Protože předpokládáme splnění polynomiálního omezení velikosti čísel, můžeme předpokládat, že $b_i = O(\log k)$ pro všechna i. Z toho plyne, že $t = O(\sum_{i=1}^{k} b_i^2 + k \sum_{i=1}^{k} b_i) = O(k \log^2 k + k^2 \log k) = O(k^2 \log k)$. □

4 Neuniformní modely

4.1 Neuniformní Turingův stroj

Ve Výpočtové složitosti I jsme ve Větě 2.5.2 dokázali, že každý jazyk, který je rozpoznatelný TS v polynomiálním čase, lze reprezentovat ve vhodném kódování pomocí posloupnosti Booleovských obvodů, které jsou konstruovatelné v polynomiálním čase od délky vstupu a, speciálně, mají polynomiální velikost. V některých situacích je vhodné uvažovat obvody polynomiální velikosti i bez požadavku, aby byly efektivně konstruovatelné. V takovém případě získáme výpočetní model, který může pro vstupy různé délky použít různé algoritmy. Tím se tento model liší jak od efektivně konstruovatelných obvodů, tak i od obvyklého TS.

Rozdíl mezi efektivně konstruovatelnými obvody a obecnými obvody lze zobecnit a rozlišujeme pak uniformní a neuniformní modely. *Uniformní model* je takový, že pro všechny délky vstupu použije stejný algoritmus. Turingův stroj a RAM v jejich základní definici jsou příklady uniformních modelů. *Neuniformní*

model může pro každou délku vstupu použít jiný algoritmus. To zvyšuje sílu těchto modelů, protože mohou například reprezentovat nerekurzivní posloupnosti funkcí. Obecné obvody jsou příkladem neuniformního modelu. Kromě těchto obvodů ale je možné neuniformní model zavést i jako variantu TS s využitím tzv. poradní (advice) funkce. Protože takto vzniklý neuniformní TS umožňuje snadněji formulovat vztah mezi uniformní a neuniformní složitostí, popíšeme tento model podrobněji.

Poradní funkce je libovolná funkce $\alpha : \mathrm{N} \to \{0,1\}^*$. Turingův stroj s poradní funkcí α je stroj, který má přidanou pomocnou pásku pouze pro čtení, na které má při výpočtu pro vstupní slovo w délky n zapsáno slovo $\alpha(n)$.

Pro libovolnou třídu jazyků T definovaných pomocí nějakého složitostního omezení TS a poradní funkci α budeme jako T/α značit třídu jazyků, pro které existuje Turingův stroj s poradní funkce α, který má jinak stejná omezení jako stroje pro třídu T, tedy čas, prostor, režim vstupní pásky (jen čtení nebo i zápis), determinismus/nedeterminismus. Označením T/poly rozumíme sjednocení T/α pro všechny funkce α, pro které je délka $\alpha(n)$ omezena polynomem od n.

Neuniformní modely, především obecné Booleovské obvody, umožňují definovat složitost i pro úlohy s omezenou délkou vstupu. Například má smysl otázka, jaký nejmenší obvod je schopen řešit problém splnitelnosti výrokových formulí délky nejvýše 1000 znaků. Neuniformní modely navíc umožňují reprezentovat pravděpodobnostní výpočty deterministicky při zachování časové a prostorové složitosti. Tuto souvislost využijeme k důkazu, že existence pravděpodobnostního polynomiálního algoritmu pro SAT by implikovala, že polynomiální hierarchie končí na hladině Σ_2. Pokud je tedy PH nekonečná, nelze SAT řešit v polynomiálním čase ani pravděpodobnostním algoritmem.

4.2 Ekvivalence obecných obvodů a P/poly

Věta 4.2.1 *Nechť L je jazyk nad abecedou Σ a nechť k je libovolné kódování $\Sigma \to \{0,1\}^d$. Pak L patří do P/poly právě tehdy, když existuje posloupnost Booleovských obvodů $\{C_n\}_{n=0}^{\infty}$ v bázi $\{\wedge, \vee, \neg\}$, která splňuje:*

1. *Velikost obvodu C_n je omezena polynomem od n.*
2. *Pro každé $n \in N$ a každé slovo $w \in \Sigma^*$, $|w| \leq n$ platí $w \in L \iff C_n(k(w)) = 1$.*

Důkaz. Nechť existuje posloupnpst obvodů C_n splňující vlastnosti z formulace věty. Jako poradní funkci zvolíme $\alpha(n) = C_n$. Turingův stroj s touto poradní funkcí má při vstupu w délky n k dispozici obvod C_n a provede simulaci výpočtu tohoto obvodu na vstup w a vydá získaný výsledek.

Nechť L je v P/poly. Vezměme libovolný TS v P/poly, který rozpoznává jazyk L, a zvolme libovolné přirozené číslo n. Zkonstruujeme obvod velikosti polynomiální v n, který vydá pro vstup w délky n hodnotu 1 právě tehdy, když $w \in L$.

Jestliže $t(n)$ je polynomiální odhad času, zkonstruujeme nejprve TS, který má jen jednu pásku, a který simuluje výpočet původního TS v čase $O(t^2(n))$. Vstupní páska, pracovní pásky i páska s poradní funkcí jsou umístěny jako stopy na pásce simulujícího stroje. Postupem použitým v důkazu Věty 2.5.2 ve Výpočtové složitosti I získáme obvod velikosti $O(t^4(n))$. Jediný rozdíl je v tom, že v tomto případě zahrnuje zkonstruovaný obvod hodnotu poradní funkce $\alpha(n)$ a tedy není obecně zkonstruovatelný efektivně. Tvrzení dokazované věty ale požaduje pouze existenci obvodu polynomiální velikosti, a proto posloupnost těchto obvodů pro všechna n dokazuje požadovanou implikaci. □

4.3 Karp-Liptonova věta

Budeme zkoumat vztah neuniformních tříd a tříd polynomiální hierarchie zavedené v Sekci 1.

Jazyk S nad abecedou A nazveme self-reducibilní, pokud existuje polynomiálně vyčíslitelná funkce f definovaná na A^* taková, že pro každé $w \in A^*$ je buď $f(w) \in \{0,1\}$ nebo $f(w) = (w_1, \ldots, w_k)$, kde všechna w_i, $i = 1, \ldots, k$ mají délku menší než w, a pro každé w platí

$$f(w) \in \{0,1\} \Rightarrow (w \in S \Leftrightarrow f(w) = 1)$$

a dále

$$f(w) = (w_1, \ldots, w_k) \Rightarrow (w \in S \iff (w_1 \in L \vee \ldots \vee w_k \in L)) \ .$$

Problém SAT je self-reducibilní. Pro slova w, která nejsou formulí, definujeme $f(w) = 0$. Pro w, které reprezentuje formuli bez proměnných, definujeme f jako hodnotu této formule. Pro formuli w, která obsahuje alespoň jednu proměnnou, definujeme $f(w)$ jako (w_1, w_2), kde w_1 a w_2 jsou formule vzniklé z w dosazením 0 a 1 za první proměnnou a redukcí vzniklého výrazu.

Problém neekvivalence read-once rozhodovacích diagramů je self-reducibilní z podobného důvodu. Pokud ani jeden z porovnávaných diagramů neobsahuje proměnné, lze test provést v konstantním čase a definujeme f na základě správné odpovědi. Pokud alespoň jeden z diagramů obsahuje proměnnou, převedeme test jejich ekvivalence na dva testy ekvivalence po dosazení hodnoty 0 a 1 za libovolnou z proměnných. Kódování diagramů musí být takové, aby délka zápisu fixací libovolné proměnné klesla.

Tvrzení 4.3.1 (Karp, Lipton, 1980) *Kdyby existoval jazyk S, který je self-reducibilní, je řešitelný polynomiálními obvody a je NP-úplný, pak $\Pi_2 = \Sigma_2$ a tedy také $\Sigma_3 = \Sigma_2$.*

Důkaz. Předpokládejme, že pro nějaký polynom $s(n)$ a každé n existuje obvod C velikosti nejvýše $s(n)$, který řeší S pro vstupy délky nejvýše n. Předpokládejme

dále, že $L \in \Pi_2$. Pak existuje relace $R \in \mathrm{P}$, která je p-vyvážená pro nějaký polynom $p_1(n)$, taková, že platí

$$w \in L \iff (\forall x_1)(\exists x_2) R(w, x_1, x_2) \ .$$

Predikát $(\exists x_2) R(w, x_1, x_2)$ je v NP a S je NP-úplný, tedy existuje p-vyčíslitelná funkce g, která splňuje $(\exists x_2) R(w, x_1, x_2) \iff g(w, x_1) \in S$. Nechť $p_2(n)$ je horní mez na délku $g(w, x_1)$ pro slova w délky nejvýše n.

Naším cílem je ukázat existenci Σ_2 formule s argumentem w, která vyjadřuje $w \in L$. Označme $n = |w|$. Požadovaná formule bude utvořena následovně. Její existenční kvantifikátor bude mimo jiné kvantifikovat též přes všechny obvody C velikosti nejvýše $s(p_2(n))$. Zbytek formule se bude skládat z konjunkce dvou podformulí. První zkontroluje, že obvod C skutečně řeší náležení do S pro slova délky nejvýše $p_2(n)$ pomocí self-reducibility S a druhá využije $C(g(w, x_1))$ ke zjištění, zda $g(w, x_1) \in S$. Pro verifikaci obvodu použijeme formuli $Verif(C, u)$, která provede simulaci výpočtu obvodu C na vstup u, jejíž výsledek označíme $C(u)$, a porovná tento výsledek buď s $f(u)$, pokud je $f(u) \in \{0, 1\}$ nebo s $C(w_1) \vee \ldots \vee C(w_k)$ pokud $f(u) = (w_1, \ldots, w_k)$. Celá formule s volnou proměnnou w má tvar

$$(\exists C) \left[(\forall u, |u| \leq p_2(n)) \, Verif(C, u) \wedge (\forall x_1) C(g(w, x_1)) \right] , \tag{5}$$

Lze ověřit, že (5) je ekvivalentní formuli $(\forall x_1)(\exists x_2) R(w, x_1, x_2)$ a tedy také $w \in L$. Převedením (5) do prenexního tvaru dostaneme formuli typu Σ_2, která vyjadřuje náležení do L. Z toho plyne $L \in \Sigma_2$. □

Věta 4.3.2 (Karp, Lipton, 1980) $\mathrm{SAT} \in \mathrm{P/poly} \Rightarrow \Pi_2 = \Sigma_2$

5 Pravděpodobnostní algoritmy

V sekcích 5.1 a 5.2 uvedeme příklady použití náhodnosti pro výpočty. V sekci 5.3 popíšeme třídu BPP, kterou tvoří úlohy řešitelné pravděpodobnostním TS v polynomiálním čase a s exponenciálně malou pravděpodobností chyby.

5.1 Využití náhodnosti pro demonstraci neizomorfnosti dvou grafů

Předpokládejme, že jsou dány grafy G_1 a G_2 na n vrcholech, o kterých předpokládáme, že nejsou izomorfní, ale je obtížné tento fakt dokázat výpočtem malé složitosti. Budeme uvažovat subjekty Prover a Verifier, které spolu mohou komunikovat. Prover má k dispozici neomezenou výpočetní kapacitu a jeho úkolem je demonstrovat důkaz neizomorfnosti G_1 a G_2 tak, aby Verifier mohl důkaz ověřit i bez složitého výpočtu. Popíšeme komunikační protokol, který toto umožňuje.

Komunikaci zahajuje Verifier, který pro zvolené číslo m vygeneruje posloupnost $r_1, \ldots, r_m \in \{1, 2\}$ a pošle Proveru grafy $H_1, \ldots, H_m$, které vytvoří tak, že H_i vznikne z grafu G_{r_i} náhodnou permutací vrcholů. Každý z grafů H_i je tedy izomorfní právě jednomu z grafů G_1 a G_2 podle hodnoty r_i. Prover provede výpočty potřebné ke zjištění, kterému z grafů G_1 a G_2 je izomorfní H_i, a sestaví posloupnost $s_1, \ldots, s_m \in \{1, 2\}$ tak, že H_i je izomorfní G_{s_i}. Prover pak pošle posloupnost $s_1, \ldots, s_m$ Verifieru. Verifier zjistí, zda $r_i = s_i$ pro všechna $i = 1, \ldots, m$, a v takovém případě důkaz přijme, jinak zamítne.

Věta 5.1.1 *Pokud jsou grafy G_1 a G_2 neizomorfní, pak Prover může provést důkaz podle popsaného protokolu tak, že jej Verifier vždy přijme. Pokud grafy nejsou izomorfní, pak Verifier důkaz přijme s pravděpodobností nejvýše 2^{-m} nezávisle na tom, jakým způsobem postupuje Prover.*

Důkaz. Pokud jsou grafy neizomorfní, je tvrzení zřejmé. Předpokládejme tedy, že G_1 a G_2 jsou izomorfní, tedy že existuje permutace vrcholů p taková, že G_2 vznikne z G_1 touto permutací vrcholů, což budeme zapisovat $G_2 = G_1^p$. Množinu všech permutací n vrcholů budeme značit S_n.

Tvrdíme, že pak jsou všechny grafy H_i vygenerovány se stejného rozdělení pravděpodobnosti, a tedy tyto grafy nenesou žádnou informaci o posloupnosti $r_1, \ldots, r_m$. Pokud $r_i = 1$, pak H_i je konstruován jako G_1^q, kde q je náhodná permutace vrcholů z S_n. Pokud $r_i = 2$, pak H_i je konstruován jako $G_2^q = (G_1^p)^q = G_1^{pq}$, kde q je náhodná permutace vrcholů z S_n. Protože S_n je grupa, jsou zobrazení $q \mapsto pq$ a $q \mapsto p^{-1}q$ navzájem inverzní. Zobrazení $q \mapsto pq$ je tedy bijekce na S_n. Z toho plyne, že pravděpodobnostní rozdělení permutací q a pq jsou stejná, a tedy jsou stejná i rozdělení H_i pro $i = 1, \ldots, m$ nezávisle na hodnotě r_i.

Protože Verifier generoval posloupnost $r_1, \ldots, r_m$ z rovnoměrného rozdělení na všech posloupnostech z $\{1, 2\}^m$ a posloupnost $s_1, \ldots, s_m \in \{1, 2\}^m$ je na ní statisticky nezávislá, je pravděpodobnost shody $r_i = s_i$ pro všechna $i = 1, \ldots, m$ rovna 2^{-m}. □

5.2 Pravděpodobnostní test neekvivalence read-once rozhodovacích diagramů

Budeme uvažovat rozhodovací diagramy, které pro vstup $x = (x_1, \ldots, x_n)$ počítají Booleovskou funkci $f : \{0, 1\}^n \to \{0, 1\}$ následovně. Výpočet začíná v počátečním uzlu. Každému nekoncovému uzlu je přiřazena některá ze vstupních proměnných x_i a uzel má následníky, které odpovídají podmínkám $x_i = 0$ a $x_i = 1$. Výpočet pokračuje do následníka, jehož podmínka je pro daný vstup splněna. Koncové uzly jsou ohodnoceny 0 nebo 1 a toto ohodnocení určuje hodnotu $f(x)$ pro daný vstup x. Rozhodovací diagram nazveme read-once, pokud každý výpočet čte každou proměnnou x_i nejvýše jednou. Funkci, kterou počítá read-once diagram P, budeme značit f^P.

Problém splnitelnosti je pro read-once diagramy snadno řešitelný. Funkce reprezentovaná read-once rozhodovacím diagramem má splňující ohodnocení právě tehdy, když v diagramu existuje cesta z počátečního do přijímajícího uzlu. Tím je problém splnitelnosti převeden na problém s,t-souvislosti grafu, který lze efektivně řešit.

Read-once diagramy P a Q nazveme ekvivalentní, pokud počítají stejnou funkci, tedy pokud $f^P = f^Q$. Ukážeme, že problém ekvivalence dvou read-once diagramů lze řešit pravděpodobnostním algoritmem v polynomiálním čase. Pokud algoritmus vydá jako výstup, že diagramy jsou neekvivaletní, je výstup dokazatelně správný. Budeme proto úlohu nazývat test neekvivalence. Algoritmus lze navíc upravit tak, že v případě neekvivalence vydá jako výstup ohodnocení proměnných $a \in \{0,1\}^n$ takové, že $f^P(a) \neq f^Q(a)$. Pokud algoritmus vydá jako výstup, že diagramy jsou ekvivaletní, může být výstup chybně, ale parametry algoritmu lze nastavit tak, že pravděpodobnost chyby je exponenciálně malá.

Pro popis pravděpodobnostního test neekvivalence, přiřadíme nejprve libovolnému read-once diagramu P polynom, který vyjadřuje funkci počítanou diagramem. Polynom definujeme indukcí přes uzly diagramu počínaje koncovými uzly. Nechť v je uzel diagramu P, který testuje proměnnou x_i a jehož 0, resp. 1, následník je v_0, resp. v_1. Jestliže uzly v_0 a v_1 již mají přiřazen polynom $f^P_{v_0}$, resp. $f^P_{v_0}$, pak definujeme

$$f^P_v = (1 - x_i) f^P_{v_0} + x_i f^P_{v_1} \ . \tag{6}$$

Jako $f^P(x_1, \ldots, x_n)$ označme polynom f^P_v, kde v je počáteční uzel diagramu. Lze snadno ověřit, že platí následující tvrzení.

Lemma 5.2.1 *Pro libovolný read-once rozhodovací diagram P a libovolný vstup $x \in \{0,1\}^n$ platí $f(x) = f^P(x)$.*

Důkaz. Dokážeme, že pro libovolný uzel diagramu v platí identita $f_v(x) = f^P_v(x)$, kde f_v je funkce počítaná read-once rozhodovacím diagramem, jehož počáteční uzel je v. Pro koncové uzly identita platí, protože $f_v(x)$ a $f^P_v(x)$ jsou stejné konstanty. Pro další uzly dokážeme identitu indukcí pomocí vztahu (6). □

Polynom $f^P_v(x)$ je definován postupem jeho výpočtu, který má složitost lineární ve velikosti diagramu P. Pokud chceme zjistit jeho koeficienty, je třeba získané výrazy roznásobit. Tím dostaneme polynom s celočíselnými koeficienty a obecně exponenciálním počtem členů. Protože polynom má pouze celočíselné koeficienty, lze jej chápat jako polynom nad libovolným tělesem T. K formulaci testu neekvivalence read-once rozhodovacích diagramů bude zapotřebí efektivní výpočet $f^P(a)$ pro libovolné $a \in T^n$.

Důsledek 5.2.2 *Nechť f^P je polynom přiřazený diagramu P a T je těleso. Výpočet $f^P(a_1, \ldots, a_n)$ lze pro libovolnou kombinaci hodnot proměnných $a_i \in T$ provést pomocí nejvýše $O(|P|)$ operací sčítání a násobení v tělese T.*

Důkaz. Pro dané ohodnocení postupně vypočteme hodnoty $f_v^P(a_1, \ldots, a_n)$ pro všechny uzly diagramu podle vztahu (6). □

Polynom nazveme multilineární, pokud se v každém jeho monomu vyskytuje každá proměnná nejvýše v první mocnině. Z konstrukce f^P vyplývá následující.

Lemma 5.2.3 *Polynom f^P je multilineární polynom.*

Důkaz. Indukcí podle postupu vytváření polynomů pro jednotlivé uzly diagramu ukážeme, že pro každý uzel v je polynom f_v multilineární. Pro koncové uzly je tvrzení zřejmé. Pro ostatní uzly plyne tvrzení ze vztahu (6) a z indukčního předpokladu pro následníky v_0 a v_1 uzlu v, protože polynomy f_{v_0} a f_{v_1} nezávisí na proměnné x_i vzhledem k tomu, že P je read-once diagram. □

Ukážeme, že ekvivalentní read-once rozhodovací diagramy definují shodné polynomy, i když postup jejich výpočtu podle diagramu může být odlišný.

Lemma 5.2.4 *Jestliže h je multilineární polynom n proměnných nad tělesem T a pro libovolné $a \in \{0,1\}^n$ platí $h(a) = 0$, pak také pro libovolné $b \in T^n$ platí $h(b) = 0$.*

Důkaz. Indukcí podle $k = 0, 1, \ldots, n$ budeme dokazovat, že $h(b) = 0$ pro $b \in T^n$ takové, že $b_i \in \{0,1\}$ pro $i \geq k+1$. Pro $k = 0$ toto platí z předpokladu. Nechť $1 \leq k \leq n$ a nechť b je takové, že $b_i \in \{0,1\}$ pro $i \geq k+1$. Uvažme $b'(t) \in T^n$ pro $t \in T$ takové, že $b'_k(t) = t$ a pro $i \neq k$ platí $b'_i(t) = b_i$. Podle indukčního předpokladu platí $h(b'(0)) = h(b'(1)) = 0$. Protože h je multilineární, je $h(b'(t))$ lineární funkcí proměnné t. Platí tedy $h(b'(t)) = 0$ pro všechna $t \in T$ a tedy i $h(b) = 0$. Tím je indukční hypotéza dokázána pro všechna $k = 0, \ldots, n$. Z platnosti pro $k = n$ plyne tvrzení lemmatu. □

Věta 5.2.5 *Jestliže f a g jsou multilineární polynomy n proměnných nad tělesem T a pro libovolné $a \in \{0,1\}^n$ platí $f(a) = g(a)$, pak také pro libovolné $b \in T^n$ platí $f(b) = g(b)$.*

Důkaz. Plyne z Lemmatu 5.2.4 pro $h(x) = f(x) - g(x)$. □

Diagramy P a Q jsou neekvivalentní, pokud existuje vstup $a \in \{0,1\}^n$, pro který je $f^P(a) \neq f^Q(a)$. Pokud je takovýchto rozlišujících vstupů málo, nemusí být náhodná volba a efektivní, protože může vést na rozlišující vstup s exponenciálně malou pravděpodobností. Pravděpodobnostní test neekvivalence je založen na tom, že na základě Věty 5.2.5 můžeme místo rozlišujících vstupů z $\{0,1\}^n$ hledat náhodně zobecněné rozlišující vstupy z T^n pro vhodné těleso T. K důkazu, že tímto způsobem lze testovat neekvivalenci pravděpodobnostně v polynomiálním čase, použijeme následující tvrzení.

Lemma 5.2.6 *Jestliže $g(x_1, \ldots, x_n)$ je libovolný nenulový multilineární polynom nad tělesem T a $M \subseteq T$ je konečná, pak existuje alespoň $(|M| - 1)^n$ kombinací hodnot $x_i \in M$, pro které je $g(x_1, \ldots, x_n) \neq 0$.*

Důkaz. Pro $n = 0$, tj. pro konstantní polynomy tvrzení platí. Dále postupujme indukcí podle počtu proměnných. Nechť g je multilineární polynom v proměnných $x_1, \ldots, x_n$. Rozdělme jeho monomy podle toho, zda obsahují nebo neobsahují x_1. Z první skupiny monomů x_1 vytkneme. Dostaneme vyjádření $g = x_1 g_1 + g_0$, kde g_1 a g_0 jsou polynomy v proměnných $x_2, \ldots, x_n$. Protože g je nenulový, je nenulový aspoň jeden z polynomů g_1 a g_0.

Nechť g_1 je nenulový. Podle indukčního předpokladu existuje aspoň $(|M| - 1)^{n-1}$ ohodnocení proměnných $x_2, \ldots, x_n$, které dávají $g_1 \neq 0$. Pro každé z nich je g lineární funkce x_1, tj. platí $g = ax_1 + b$ pro nějaké $a \neq 0$ a nějaké b. Pak pro každou volbu $x_1 \in M \setminus \{-\frac{b}{a}\}$ je $g \neq 0$. Každé z uvažovaných $(|M| - 1)^{n-1}$ ohodnocení proměnných $x_2, \ldots, x_n$ jsme tedy doplnili alespoň $|M| - 1$ způsoby na ohodnocení všech proměnných, které splňuje $g \neq 0$.

Nechť $g_1 = 0$ a g_0 je nenulový. Pak podle indukčního předpokladu existuje aspoň $(|M|-1)^{n-1}$ ohodnocení proměnných $x_2, \ldots, x_n$, které dávají $g_0 \neq 0$. Každé z nich lze doplnit libovolnou volbou $x_1 \in M$ na ohodnocení všech proměnných, které dává $g \neq 0$. V tomto případě tedy máme dokonce alespoň $|M|(|M| - 1)^{n-1}$ potřebných ohodnocení. □

Příklad. Jestliže $g(x_1, \ldots, x_n) = x_1 \ldots x_n$ a $M \subseteq T$, pak počet kombinací hodnot z M, pro které je g nenulový, je buď $|M|^n$, pokud $0 \notin M$, nebo $(|M| - 1)^n$, pokud $0 \in M$. V obecném případě tedy dolní mez z lemmatu nelze zlepšit.

Lemma 5.2.6 umožňuje alternativní důkaz Lemmatu 5.2.4, když použijeme $M = \{0, 1\}$. Kdyby existovalo ohodnocení $a \in T^n$, pro které je $h(a) \neq 0$, pak podle Lemmatu 5.2.6 pro $M = \{0, 1\}$ by muselo existovat alespoň jedno ohodnocení $b \in \{0, 1\}^n$, pro které $h(b) \neq 0$, což by byl spor s předpokladem.

Pokud nalezneme zobecněné rozlišující ohodnocení, lze efektivně najít i rozlišující ohodnocení z $\{0, 1\}^n$.

Věta 5.2.7 *Jestliže g je multilineární polynom n proměnných, umíme jej efektivně vyhodnotit a známe některé ohodnocení $a \in T^n$, pro které platí $g(a) \neq 0$, pak lze efektivně najít ohodnocení $b \in \{0, 1\}^n$ takové, že $g(b) \neq 0$.*

Důkaz. Protože $g(x_1, a_2, \ldots, a_n)$ je lineární funkce proměnné x_1, pak z toho, že je nenulová pro $x_1 = a_1$ plyne, že je nenulová pro alespoň jednu z hodnot $x_1 = 0$ a $x_1 = 1$. Změníme a_1 na $a'_1 = 0$ nebo $a'_1 = 1$ tak, aby $g(a'_1, a_2, \ldots, a_n) \neq 0$. Pak opakujeme stejný postup s a_i pro $i = 2, \ldots, n$. Výsledkem je ohodnocení $(a'_1, \ldots, a'_n) \in \{0, 1\}^n$ takové, že $g(a'_1, \ldots, a'_n) \neq 0$. □

Uveďme si ještě následující tvrzení, které není potřeba pro test neekvivalence, ale ilustruje význam polynomu f^P nad tělesem racionálních čísel.

Věta 5.2.8 Nechť P je read-once rozhodovací diagram pro funkci f proměnných $x_1, \ldots, x_n$. Pak $f^p(x_1, \ldots, x_n)$ je roven polynomu, který vznikne následujícím způsobem

(i) Každé hraně P, která odpovídá testu $x_i = 1$ přiřaďme výraz x_i a každé hraně odpovídající testu $x_i = 0$ přiřaďme výraz $1 - x_i$.

(ii) Každé cestě v P přiřaďme součin výrazů pro její jednotlivé hrany.

(iii) Diagramu P přiřadíme součet součinů přiřazených všem cestám v P z počátečního uzlu do přijímajícího uzlu.

Následující lemma dává možnost interpretovat polynom f^p jako pravděpodobnost určitého typu událostí.

Lemma 5.2.9 *Nechť jsou hodnoty $x_i \in \{0, 1\}$ voleny náhodně, nezávisle a tak, že $\Pr(x_i = 1) = p_i$. Pak $\Pr(f(x_1, \ldots, x_n) = 1) = f^p(p_1, \ldots, p_n)$.*

Důkaz. Tvrzení plyne z Věty 5.2.8 a z toho, že pravděpodobnost, že náhodný vstup projde po dané cestě, je právě rovna součinu ohodnocení jejích hran. Průchod po různých cestách jsou náhodné jevy, které se vylučují. Proto je pravděpodobnost disjunkce těchto jevů rovna součtu jejich pravděpodobností.

Tvrzení je možné dokázat také indukcí přes uzly diagramu pomocí vztahu (6). □

Lemma 5.2.9 umožňuje zjistit počet splňujících ohodnocení pro daný diagram, protože počet splňujících ohodnocení funkce f je právě $f^p(1/2, \ldots, 1/2)2^n$. Algoritmus provede polynomiální počet operací s čísly, jejichž délka zápisu je $O(n)$. Jeho složitost je tedy polynomiální v n.

Nyní popišme jeden krok pravděpodobnostního algoritmu pro testování neekvivalence read-once diagramů. Jednoduchou podobu celého algoritmu dostaneme opakováním tohoto základního kroku nebo zvětšením použité množiny M, přičemž spolehlivost výsledku roste s počtem opakování nebo s velikostí M a tedy také s počtem použitých náhodných bitů. Existuje složitější postup, který pracuje s racionálními aproximacemi odmocnin prvočísel, pro který lze spolehlivost výsledku zvyšovat za cenu nárůstu složitosti deterministické části algoritmu, tedy bez narůstání počtu potřebných náhodných bitů. Tento algoritmus vystačí s n náhodnými bity, ale není předmětem tohoto textu.

Následující algoritmus pracuje s libovolným tělesem T a předpokládá přesnou aritmetiku v tomto tělese. Pro dosažení efektivního postupu je možné pracovat s konečným tělesem, například se Z_q pro prvočíslo q.

Test neekvivalence.
Vstup jsou dva diagramy P_1 a P_2 a $M \subseteq T$.
Nechť f_1^p a f_2^p jsou polynomy přiřazené P_1 a P_2 v uvedeném pořadí.

Vyberme náhodně a nezávisle hodnoty $a_1, \ldots, a_n \in M$
z rovnoměrného rozdělení na M.
Pro vybrané hodnoty proměnných vyčísleme polynomy f_1^p a f_2^p.
Je-li $f_1^p(a_1, \ldots, a_n) \neq f_2^p(a_1, \ldots, a_n)$, pak return$(a_1, \ldots, a_n)$,
jinak return$(\emptyset)$.

Je-li výstup algoritmu ohodnocení $a_1, \ldots, a_n$, které odlišuje polynomy f_1^p a f_2^p, pak jsou vstupní diagramy neekvivalentní a získané ohodnocení budeme nazývat důkaz (svědek) neekvivalence P_1 a P_2. Na základě Věty 5.2.7 je pak možné najít také ohodnocení z množiny $\{0, 1\}$, které diagramy odlišuje. Jestliže je výstup algoritmu $\emptyset$, mohou být diagramy ekvivalentní i neekvivalentní. Platí ale následující tvrzení.

Věta 5.2.10 *Nechť P_1 a P_2 jsou dva read-once rozhodovací diagramy s n proměnnými. Pak platí následující.*
(i) Jsou-li P_1 a P_2 ekvivalentní, pak algoritmus vždy vydá odpověď $\emptyset$.
(ii) Pokud jsou neekvivalentní, pak pravděpodobnost, že algoritmus najde důkaz neekvivalence, je alespoň $1 - n/|M|$.

Důkaz. Jestliže jsou P_1 a P_2 ekvivalentní, pak $f_1^p = f_2^p$ a algoritmus tedy vydá hodnotu $\emptyset$.

Jestliže P_1 a P_2 nejsou ekvivalentní, pak $f_1^p - f_2^p$ je nenulový multilineární polynom, protože nabývá nenulové hodnoty již pro některou kombinaci $a_i \in \{0, 1\}$. Pravděpodobnost, že algoritmus najde důkaz neekvivalence, vypočteme jako poměr počtu příznivých případů ku počtu všech případů. Počet všech výběrů hodnot proměnných je $|M|^n$. Podle Lemmatu 5.2.6 je počet příznivých případů alespoň $(|M| - 1)^n$. Poměr je tedy alespoň

$$\frac{(|M| - 1)^n}{|M|^n} = \left(1 - \frac{1}{|M|}\right)^n .$$

Protože $(1 - x)^n \geq 1 - nx$, dostaneme, že

$$\left(1 - \frac{1}{|M|}\right)^n \geq 1 - \frac{n}{|M|} .$$

Tím je odhad dokázán. □

Nechť M je libovolná podmnožina T velikosti $2n$. Pokud je charakteristika T alespoň $2n$ nebo 0, lze použít $M = \{0, 1, \ldots, 2n - 1\}$. Pak v předchozím tvrzení dostaneme odhad pravděpodobnosti $1/2$. Přesnější výpočet by ukázal, že pro velké n je pravděpodobnost nalezení důkazu neekvivalence alespoň $\left(1 - \frac{1}{2n}\right)^n \approx \mathrm{e}^{-1/2} \approx 0.60653$.

Pravděpodobnost chyby lze snížit na exponenciálně malou hodnotu, pokud algoritmus několikrát opakujeme pro (statisticky) nezávislé hodnoty náhodných

bitů nebo pokud dostatečně zvětšíme množinu M. Z hlediska počtu potřebných náhodných bitů je použití větší množiny M efektivnější postup zvýšení spolehlivosti než opakování.

Věta 5.2.11 *Nechť $k \geq 1$ je přirozené číslo. Pokud $|M| = 2^k n$, pak pravděpodobnost nalezení důkazu neekvivalence pomocí výše popsaného algoritmu je alespoň*

$$1 - \frac{1}{2^k} \ .$$

Důkaz. Z Věty 5.2.10 plyne

$$\left(1 - \frac{1}{2^k n}\right)^n \geq 1 - \frac{1}{2^k} \ .$$

□

Poznamenejme, že k reprezentaci jednoho prvku M stačí v tomto případě $\log_2 |M| = k + \log_2 n$ bitů. K provedení algoritmu je tedy v tomto případě potřeba generovat $n \log_2 |M| = (k + \log_2 n)n$ náhodných bitů.

5.3 Třída BPP

Pravděpodobnostní Turingův stroj bez omezení složitosti lze zavést například následujícím způsobem.

Definice 5.3.1 *Pravděpodobnostní TS (PTS) je TS s přidanou páskou, na níž je před zahájením výpočtu pro vstup w délky n zapsána posloupnost náhodných bitů délky $s(n)$. PTS může při svém výpočtu využívat tyto náhodné bity, proto může mít pro jedno vstupní slovo w více výpočtů, ale žádný výpočet nesmí na pásce s náhodnými bity dojít za poslední náhodný bit, tedy nemůže testovat jeho délku $s(n)$.*

V dalším se omezíme na takové PTS, které pracují v polynomiálním čase a tedy $s(n)$ může mít polynomiálně omezenou délku.

Jak je vidět z definice, lze PTS uvažovat jako speciální případ nedeterministického TS. Stejně jako nedeterministický stroj, činí i PTS během výpočtu rozhodnutí, která nezávisí jen na vstupním slově, ale ještě na dalších pomocných bitech. Pravděpodobnostní a nedeterministický stroj se liší kvantifikací požadovaného počtu kombinací pomocných bitů, pro které dojde k přijetí vstupního slova.

Jestliže M je PTS, w je vstup a r je posloupnost náhodných bitů, které jsou použity při výpočtu, pak výsledek výpočtu budeme značit $M(w, r)$. Pokud je možné výsledek $M(w, r)$ interpretovat jako přijetí nebo nepřijetí, pak budeme jako $R(w, r)$ značit predikát, který je splněn pro takové kombinace w, r, pro které

$|r| = s(|w|)$ a výsledek $M(w, r)$ znamená přijetí. Pravděpodobnost chyby budeme vyjadřovat pomocí pravděpodobnosti, že platí $R(w, \tilde{r})$, kde $\tilde{r}$ je náhodná kombinace bitů použitých při výpočtu. Protože předpokládáme, že délka $\tilde{r}$ je konečná a závisí jen na délce vstupu w, je pravděpodobnostní prostor pro náhodný jev $R(w, \tilde{r})$ diskrétní. Pravděpodobnost $R(w, \tilde{r})$ tedy můžeme vyjádřit jako podíl počtu kombinací náhodných bitů na vstupu, které vedou k přijetí, a všech možných kombinací těchto bitů.

Pro definici pravděpodobnostního rozpoznávání jazyka L pomocí PTS použijeme libovolné reálné funkce $\alpha(n)$ a $\beta(n)$ s hodnotami v $[0, 1]$. Obvykle jsou tyto funkce buď konstantní nebo při $n \to \infty$ platí $\alpha(n) \to 0$ a $\beta(n) \to 1$.

Definice 5.3.2 Jestliže $0 \leq \alpha(n) < \beta(n) \leq 1$ jsou reálné funkce, pak řekneme, že M je $(\alpha(n), \beta(n))$-PTS pro jazyk L, jestliže platí:

$$\begin{aligned} w \notin L &\Rightarrow \Pr(R(w, \tilde{r})) \leq \alpha(|w|), \\ w \in L &\Rightarrow \Pr(R(w, \tilde{r})) \geq \beta(|w|). \end{aligned}$$

Pro slovo w délky n je pravděpodobnost chybného výsledku v případě $w \in L$ nejvýše $1 - \beta(n)$ a v případě $w \notin L$ nejvýše $\alpha(n)$. V obou případech je tedy pravděpodobnost chybného výsledku nejvýše $\max\{\alpha(n), 1 - \beta(n)\}$. Pokud nás zajímá jen tato celková pravděpodobnost chyby, pak lze použít $(\varepsilon(n), 1 - \varepsilon(n))$-PTS, kde $0 \leq \varepsilon(n) < \frac{1}{2}$ je libovolná reálná funkce.

Definice 5.3.3 Definujme třídy RP, co-RP a BPP tak, že pro každý jazyk L platí:
(i) $L \in$ RP $\iff$ existuje polynomiální $(0, \frac{1}{2}) -$ PTS pro L;
(ii) $L \in$ co-RP $\iff \setminus L \in$ RP;
(iii) $L \in$ BPP $\iff$ existuje polynomiální $(\frac{1}{3}, \frac{2}{3}) -$ PTS pro L.

Poznamenejme, že z $(\alpha(n), \beta(n))$-PTS pro jazyk L lze záměnou přijetí a nepřijetí ve výstupu získat $(1-\beta(n), 1-\alpha(n))$-PTS pro doplněk L. Proto lze co-RP ekvivalentně charakterizovat jako třídu jazyků, pro které existuje $(\frac{1}{2}, 1)$-PTS.

Stroje pro jazyky z RP jsou dosti podobné nedeterministickému TS. Pokud slovo do přijímaného jazyka nepatří, pak neexistuje přijímající výpočet. Pokud do jazyka patří, pak existuje přijímající výpočet. Je tedy RP $\subseteq$ NP. Podobně jako v NP, i v RP platí, že pokud stroj vstup přijme, pak máme jistotu, že vstup do jazyka patří. Pokud stroj vstup nepřijme, pak slovo do jazyka patřit může i nemusí. Narozdíl od NP však v RP požadujeme, aby platilo, že pokud existuje aspoň jeden přijímající výpočet, pak jich existuje mnoho. To způsobuje, že narozdíl od NP, která obsahuje jazyky, jejichž přijímaní se považuje za výpočetně náročné, jsou jazyky z RP považovány za jazyky, jejichž přijímání lze provést

efektivně. Důvod vyplyne z následujících vět, kdy ukážeme, že pravděpodobnost, že PTS přijímající jazyk z RP vydá chybný výsledek, lze snížit na exponenciálně malou hodnotu.

Pro algoritmy ve třídě BPP lze pravděpodobnost chybného výsledku snížit na exponenciálně malou hodnotu, ale přijetí ani nepřijetí vstupu nezaručuje jistou odpověď.

Výše uvedená tvrzení nyní dokážeme pomocí vět o zesilování pravděpodobnosti správné odpovědi. Obecně se toto zesilování udělá tak, že výchozí algoritmus zopakujeme několikrát pro nezávisle volené náhodné bity. Při důkazech zmíněných vět pak je především potřeba odhadnout počet opakování, který je zapotřebí k dosažení určité spolehlivosti výsledku. Omezujeme se na ty případy, kdy je tento počet opakování polynomiální v délce vstupu, tj. kdy získaný spolehlivější algoritmus je stále ještě polynomiální.

Věta 5.3.4 *Nechť $q(n)$ je libovolný polynom. Pak pro každý jazyk L platí:*
(i) Jestliže existuje $\left(0, \frac{1}{q(n)}\right)$-PTS *pro L, pak $L \in$ RP;*
(ii) Jestliže $L \in$ RP, pak existuje $\left(0, 1 - \left(\frac{1}{2}\right)^{q(n)}\right)$-PTS *pro L.*

Důkaz. K důkazu (i) i (ii) použijeme následující obecnou konstrukci. Nechť M je $(0, \varepsilon(n))$-PTS pro nějaký jazyk L a nechť $k(n)$ je nějaká funkce z přirozených čísel do přirozených čísel. Uvažme stroj M', který na vstupu w délky n zopakuje $k(n)$ krát výpočet stroje M a vstup přijme právě tehdy, když alespoň jedno z opakování vstup přijalo. Tvrdíme, že M' je $(0, 1 - e^{-\varepsilon(n)k(n)})$-PTS pro stejný jazyk.

Je-li $w \notin L$, pak žádné opakování vstup nepřijme, tj. M' přijme vstup s pravděpodobností 0. Vypočtěme dolní odhad pravděpodobnosti, že M' vstup přijme, jestliže $w \in L$. Přesněji řečeno, odvodíme horní odhad na pravděpodobnost nepřijetí v tomto případě. Označme $n = |w|$. Pro jedno opakování je pravděpodobnost nepřijetí podle předpokladu nejvýše $1 - \varepsilon(n)$. Pro $k(n)$ nezávislých opakování je pravděpodobnost, že vstup bude vždy zamítnut, právě součin pravděpodobností, že bude zamítnut v jednotlivých opakováních. Tento součin je nejvýše $(1 - \varepsilon(n))^{k(n}$. Použijeme-li nerovnost $1 - x \leq e^{-x}$, která platí pro každé reálné x, dostaneme, že pravděpodobnost zamítnutí je pro M' nejvýše

$$(1 - \varepsilon(n))^{k(n)} \leq e^{-\varepsilon(n)k(n)}.$$

Pravděpodobnost přijetí je tedy alespoň $1 - e^{-\varepsilon(n)k(n)}$, jak bylo požadováno.

K důkazu (i) použijeme $\varepsilon(n) = \frac{1}{q(n)}$ a $k(n) = q(n)$. Protože platí $1 - e^{-\varepsilon(n)k(n)} = 1 - \frac{1}{e} \geq \frac{1}{2}$, je (i) dokázáno.

K důkazu (ii) použijeme $\varepsilon(n) = \frac{1}{2}$ a $k(n) = 2q(n)$. Protože platí $1 - e^{-\varepsilon(n)k(n)} = 1 - e^{-q(n)} \geq 1 - \left(\frac{1}{2}\right)^{q(n)}$, je (ii) dokázáno. □

Věta 5.3.5 *Nechť $q(n)$ je libovolný polynom. Pak pro každý jazyk L platí:*
(i) Jestliže $L \in$ BPP, pak existuje $\left(\frac{1}{2^{q(n)}}, 1 - \frac{1}{2^{q(n)}}\right)$-PTS pro L.
(ii) Jestliže $\beta(n) - \alpha(n) \geq \frac{1}{q(n)}$ a jestliže existuje $(\alpha(n), \beta(n))$-PTS pro L, pak $L \in$ BPP;

Důkaz. Použijeme tzv. Černovovu nerovnost. Nechť $X \in \{0, 1\}$ je náhodná veličina a nechť $\Pr(X = 1) = p$ a $\Pr(X = 0) = 1 - p$ pro nějaké $p \in [0, 1]$. Protože $X \in \{0, 1\}$, platí také $\mathrm{E}X = \Pr(X = 1) = p$, kde $\mathrm{E}X$ označuje střední hodnotu X. Nechť X_i pro $i = 1, 2, \ldots, m$ jsou nezávislé realizace veličiny X. Pak pro každé $\varepsilon \geq 0$ platí

$$\Pr\left(\sum_{i=1}^{m} X_i \geq (p + \varepsilon)m\right) \leq \mathrm{e}^{-2\varepsilon^2 m}$$

a

$$\Pr\left(\sum_{i=1}^{m} X_i \leq (p - \varepsilon)m\right) \leq \mathrm{e}^{-2\varepsilon^2 m}.$$

Dokažme nejprve (i). Nechť M je $(1/3, 2/3)$-PTS pro nějaký jazyk L. Zkonstruujeme TS M', který pro slovo w délky n provede simulaci $13q(n)$ nezávislých opakování M a vydá většinový výsledek z těchto opakování jako svůj výstup. Dokážeme, že M' splňuje podmínku z tvrzení (i). Výsledek jednotlivých opakování označme jako X_i pro $i = 1, \ldots, m$, kde $m = 13q(n)$. Jestliže $w \notin L$, pak $\Pr(X = 1) = p \leq 1/3$. Pravděpodobnost chybného přijetí alespoň $m/2$ opakováními splňuje nerovnost

$$\Pr\left(\sum_{i=1}^{m} X_i \geq \frac{m}{2}\right) \leq \Pr\left(\sum_{i=1}^{m} X_i \geq \left(p + \frac{1}{6}\right) m\right) \leq \mathrm{e}^{-m/18}$$

Protože platí $\mathrm{e}^{-13/18} < 1/2$, dostaneme

$$\Pr\left(\sum_{i=1}^{m} X_i \geq \frac{m}{2}\right) \leq \mathrm{e}^{-m/18} < \frac{1}{2^{q(n)}}\ .$$

Jestliže $w \in L$, pak $\Pr(X = 1) = p \geq 2/3$. Pravděpodobnost přijetí alespoň $m/2$ opakováními odhadneme jako doplněk pravděpodobnosti nepřijetí. Z Černovovy nerovnosti plyne

$$\Pr\left(\sum_{i=1}^{m} X_i \leq \frac{m}{2}\right) \leq \Pr\left(\sum_{i=1}^{m} X_i \leq \left(p - \frac{1}{6}\right) m\right) \leq \mathrm{e}^{-m/18}$$

S využitím stejného numerického odhadu jako výše dostaneme

$$\Pr\left(\sum_{i=1}^{m} X_i \leq \frac{m}{2}\right) \leq \mathrm{e}^{-m/18} < \frac{1}{2^{q(n)}}$$

a tedy celkově

$$\Pr\left(\sum_{i=1}^{m} X_i \geq \frac{m}{2}\right) = 1 - \Pr\left(\sum_{i=1}^{m} X_i < \frac{m}{2}\right) \geq 1 - \frac{1}{2^{q(n)}} \ .$$

Tím je (i) dokázáno.

Nechť M je $(\alpha(n), \beta(n))$-PTS pro nějaký jazyk L. Pro důkaz (ii) zkonstruujeme M', který pro libovolný vstup w několikrát nezávisle zopakuje M. Počet opakování $N(w)$ zvolíme v tomto případě jako $N(w) = 4q^2(n)$. Výsledek i-tého opakování M označme jako X_i, kdy nepřijetí budeme reprezentovat číslem 0 a přijetí číslem 1. Všechna X_i jsou realizace téže náhodné veličiny X. Stroj M' přijme vstup právě tehdy, jestliže $\sum_{i=1}^{N(w)} X_i \geq \frac{\alpha(|w|)+\beta(|w|)}{2} N(w)$.

S pomocí Černovovy nerovnosti nyní odhadněme pravděpodobnost, že M' přijme slovo $w \notin L$ a slova $w \in L$. Pro jednoduchost označíme $m = N(w)$ a místo $\alpha(n)$ a $\beta(n)$ budeme psát α a β. V případě $w \notin L$ je $\mathrm{E}X = \Pr(X = 1) \leq \alpha$. Je tedy

$$\Pr\left(\sum_{i=1}^{m} X_i \geq \frac{\alpha+\beta}{2} m\right) = \Pr\left(\sum_{i=1}^{m} X_i \geq \left(\alpha + \frac{\beta-\alpha}{2}\right) m\right) \leq \mathrm{e}^{-\frac{1}{2}(\beta-\alpha)^2 m} \ .$$

Vypočtěme ještě pravděpodobnost zamítnutí pro $w \in L$. V tomto případě je $\Pr(X = 1) \geq \beta$. Je tedy

$$\Pr\left(\sum_{i=1}^{m} X_i < \frac{\alpha+\beta}{2} m\right) = \Pr\left(\sum_{i=1}^{m} X_i < \left(\beta - \frac{\beta-\alpha}{2}\right) m\right) \leq \mathrm{e}^{-\frac{1}{2}(\beta-\alpha)^2 m} \ .$$

V případě $w \in L$ je tedy pravděpodobnost přijetí alespoň $1 - e^{-\frac{1}{2}(\beta-\alpha)^2 m}$. Protože předpokládáme, že $\beta(n) - \alpha(n) \geq \frac{1}{q(n)}$ a $m = 4q^2(n)$, dostaneme

$$\mathrm{e}^{-\frac{1}{2}(\beta(n)-\alpha(n))^2 4q^2(n)} \leq \mathrm{e}^{-\frac{1}{2q^2(n)} 4q^2(n)} = e^{-2} \leq \frac{1}{3}.$$

Stroj M' je tedy $(\frac{1}{3}, \frac{2}{3})$-PTS, jak bylo zapotřebí pro (ii). □

K odvození důsledku Věty 5.3.5 pro vztah BPP a P/poly zaveďme následující pojem.

Definice 5.3.6 Nechť L je jazyk z BPP a nechť $R(w, r)$ je relace popisující přijetí w pomocí některého PTS pro L. Pak slovo r nazveme dobrá nápověda pro všechna slova délky n v jazyce L, jestliže pro každé slovo w délky n platí $w \in L \iff R(w, r)$.

Všimněme si, že známe-li nějakou dobrou nápovědu r_n pro délku n, pak můžeme pravděpodobnostní algoritmus použít pro slova délky n deterministicky, protože pro libovolné slovo w délky n stačí zjistit pouze to, zda $R(w, r_n)$.

Věta 5.3.7 *Jestliže L je z BPP, tedy také z RP nebo z co-RP, pak pro něj existuje relace R v* P *tak, že pro každé dostatečně veliké n existuje slovo r_n délky polynomiální v n, které je dobrou nápovědou pro všechna slova délky n v jazyce L.*

Důkaz. Zvolme $q(n) = n^2$. Podle bodu (ii) z Věty 5.3.5 dostaneme, že existuje polynomiální $(\varepsilon(n), 1-\varepsilon(n))$-PTS pro L, kde $\varepsilon(n) = \frac{1}{2^{n^2}}$. Nechť $R(w,r)$ popisuje tento PTS a nechť $s(n)$ je délka použitých posloupností náhodných bitů r využitých při jeho výpočtu. Označme jako V_n množinu všech slov v abecedě $\{0,1\}$ délky $s(n)$. Pak pro každé slovo w délky n označme D_w množinu "dobrých" r, což jsou slova $r \in V_n$, pro která platí $w \in L \Leftrightarrow R(w,r)$, tedy jsou to ty kombinace náhodných bitů, které vedou ke správné odpovědi.

Nechť w je slovo délky n. Označme $S_w = V_n \setminus D_w$. Slova z S_w jsou tedy ty kombinace náhodných bitů, které vedou ke špatné odpovědi.

Protože vycházíme z $(\varepsilon(n), 1-\varepsilon(n))$-PTS pro L, dostaneme následující. Je-li $w \in L$, pak

$$|D_w| \geq (1-\varepsilon(n))|V_n|.$$

a tedy

$$|S_w| = |V_n \setminus D_w| \leq \varepsilon(n)|V_n| \ .$$

Je-li $w \notin L$, pak platí

$$|S_w| \leq \varepsilon(n)|V_n|,$$

stejně jako v případě $w \in L$.

Všimněme si, že k důkazu věty stačí ukázat, že pro každou dost velikou délku n existuje slovo r, které patří do průniku D_w pro všechna slova w délky n. Toto slovo pak bude "dobrou nápovědou" pro danou délku n. Podle de Morganových zákonů platí

$$\bigcap_{|w|=n} D_w = V_n \setminus \bigcup_{|w|=n} S_w. \tag{7}$$

Klíčovým krokem důkazu je, že s využitím výše uvedeného odhadu dostaneme, že velikost S_w je natolik malá ve srovnání s V_n, že i sjednocení všech těchto množin pro všechna slova w délky n má méně prvků než V_n, a tedy pravá strana (7) je neprázdná.

Pro každé slovo w délky n je $|S_w| \leq \varepsilon(n)|V_n|$. Jestliže abeceda jazyka L je Σ, pak je celkem $|\Sigma|^n$ slov w délky n. Velikost sjednocení v pravé straně (7) je tedy nejvýše $|\Sigma|^n \varepsilon(n)|V_n| = 2^{n\log|\Sigma| - n^2}|V_n|$. Protože pro každé dostatečně veliké n je $2^{n\log|\Sigma|-n^2} < 1$, zbyde po odečtení uvažovaného sjednocení od V_n stále neprázdná množina. □

Dokázaná věta má následující důsledek pro jazyky z BPP.

Věta 5.3.8 *Jestliže L je z BPP, pak L patří do P/poly.*

Důkaz. Pro jednoduchost můžeme předpokádat, že L je v abecedě $\{0,1\}$. Protože L je v BPP, plyne z předchozí věty, že existuje polynomiálně vyčíslitelná relace R a pro každé dostatečně veliké n slovo r_n tak, že pro libovolné w délky n platí $w \in L \iff R(w, r_n)$.

Již dříve jsme se zmiňovali, že libovolný TS, který pracuje v polynomiálním čase, lze simulovat obvody polynomiální velikosti. Použijeme to na stroj, který rozpoznává relaci R. Dostaneme tak posloupnost obvodů C_k pro $k = 1, 2, \ldots$ tak, že obvod C_k rozpoznává pro slova z délky k, zda $z \in R$. Zde předpokládáme nějaké zakódování symbolů abecedy relace R do abecedy $\{0,1\}$.

Posloupnost obvodů pro jazyk L dostaneme tak, že pro slova w délky n zvolíme za k délku slov $\langle w, r_n\rangle$ a uvážíme obvod C_k. Jestliže na vstup obvodu C_k dáme $\langle w, r_n\rangle$ pro libovolné slovo w délky n, pak jako výstup dostaneme odpověď, zda $w \in L$. Jestliže v tomto obvodu budeme r_n považovat za konstantní vstup, jinak řečeno, budeme r_n považovat za součást obvodu, dostaneme obvod C'_n, jehož vstupem je pouze slovo w. Pro každé dostatečně veliké n rozpoznává obvod C'_n slova $w \in L$ délky n. Zbývá konečný počet délek n, pro které Věta 5.3.7 nezaručuje existenci r_n, a tedy nemáme zatím ani obvod C'_n. Pro tyto délky n zkonstruujeme obvod C'_n libovolně.

Protože zkonstruované obvody mají velikost polynomiální ve velikosti jejich vstupu, je věta dokázána. □

Ve spojení s Karp-Liptonovou větou dostaneme jako důsledek, že SAT nepatří do BPP, pokud je polynomiální hierarchie nekonečná. Kdyby SAT patřil do BPP, pak patří také do P/poly. Z toho podle Důsledku 4.3.2 plyne $\Pi_2 = \Sigma_2$ a polynomiální hierarchie by byla rovna své druhé hladině.

Důsledek 5.3.9 *Jestliže SAT patří do BPP, pak* $\Pi_2 = \Sigma_2$.

Uvedený důsledek má význam především v situaci, kdy předpokládáme, že P $\neq$ BPP. Pokud bychom předpokládali, že P = BPP, pak je SAT $\in$ BPP právě tehdy, když SAT $\in$ P, a následující triviální tvrzení

$$\mathrm{P} = \mathrm{BPP} \Rightarrow (\mathrm{SAT} \in \mathrm{BPP} \Rightarrow \mathrm{P} = \mathrm{NP})$$

zesiluje Důsledek 5.3.9.

Pro test neekvivalence read-once rozhodovacích diagramů, o kterém víme, že je self-reducibilní a má polynomiální obvody, dostaneme z Věty 4.3.1, že není NP-úplný, pokud je polynomiální hierarchie neomezeně rostoucí.