1 Operace s OBDD

Minimalizací OBDD budeme rozumět nalezení minimálního π-OBDD na základě libovolného π-OBDD. Syntézou π-OBDD pomocí k-ární spojky α budeme rozumět nalezení π-OBDD pro funkci $\alpha(f_1, \ldots, f_k)$, na základě vstupních π-OBDD pro f_i pro $i = 1, \ldots, k$. Operaci syntézy uvádíme v této obecné formě, protože v technických aplikacích se používá spojka $\alpha(x, y, z) = \mathrm{ite}(x, y, z) =$ if x then y else z. Pro fixované pořadí π lze uvedené operace provést v polynomiálním čase.

1.1 Algoritmus minimalizace OBDD

V předchozím textu jsme dokázali následující dvě věty, které charakterizují velikost minimálního π-OBDD.

Věta 1.1 *Pro každé* π-OBDD *pro* f*, jehož všechny uzly jsou dosažitelné ze vstupního uzlu, je množina subfunkcí počítaných v jeho uzlech rovna* $S(f, \pi)$*.*

Věta 1.2 *Pro libovolnou funkci* f *a uspořádání* π *je*

$$\pi\text{-OBDD}(f) = |S(f, \pi)| \ .$$

Navíc, π-OBDD *minimální velikosti pro* f *je až na izomorfizmus určeno jednoznačně.*

Postupně ukážeme, že π-OBDD, které je minimální pro danou funkci, lze charakterizovat syntaktickou podmínkou, tedy podmínkou týkající se uzlů a hran OBDD, nikoli funkcí, které uzly počítají.

Věta 1.3 *Libovolné* π-OBDD *je minimální pro dané uspořádání* π *právě tehdy, když každé dva jeho různé uzly počítají různé funkce.*

Důkaz. Z Vět 1.1 a 1.2 plyne, že všechna minimální π-OBDD jsou izomorfní a existuje bijekce mezi jejich uzly a množinou subfunkcí $S(f, \pi)$. Minimální OBDD tedy nemůže obsahovat dva uzly počítající stejnou funkci.

Pokud π-OBDD neobsahuje dva uzly počítající tutéž funkci, pak z Věty 1.1 plyne, že obsahuje právě $|S(f, \pi)|$ uzlů a je tedy minimální. □

Věta 1.4 *Libovolné* π-OBDD *je minimální pro dané uspořádání* π *právě tehdy, když splňuje následující tři podmínky.*

(i) Všechny jeho uzly jsou dosažitelné z počátečního uzlu.

(ii) Žádný uzel nemá oba následníky stejné.

(iii) Neexistují v něm dva různé uzly, které testují tutéž proměnnou a mají stejného 0-následníka i 1-následníka.

Důkaz. Uvedené podmínky jsou zřejmě nutné. Ukážeme, že uvedené podmínky implikují, že každé dva uzly počítají různé subfunkce. Spolu s Větou 1.3 to implikuje, že jsou postačující.

Nechť π-OBDD splňuje (i), (ii) a (iii), ale obsahuje různé uzly, které počítají tutéž funkci. Zvolme některou takovou dvojici uzlů (u, v), která má minimální vzdálenost od koncových uzlů v následujícím smyslu. Požadujeme, že neexistuje žádná dvojice uzlů (u', v'), které také počítají tutéž funkci, která by byla různá od (u, v) a taková, že z u do u' a z v do v' vede cesta (za cestu považujeme i cestu nulové délky). Protože je graf konečný, požadovaná minimální dvojice uzlů (u, v) existuje.

Předpokládejme nejprve, že u a v testují různé proměnné. Nechť například u testuje proměnnou nižšího indexu v π. Pak u počítá subfunkci, která nezávisí podstatně na testované proměnné a tedy oba jeho následníci počítají tutéž funkci jako u. Protože platí (ii), má u dva různé následníky a alespoň jeden z nich je různý od v a tedy tvoří s uzlem v dvojici, která je ve sporu s volbou (u, v).

Předpokládejme nyní, že u a v testují tutéž proměnnou. Označme jako u_0, u_1, v_0, v_1 jejich 0-následníky a 1-následníky. Protože Q splňuje (iii), platí buď $u_0 \neq v_0$ nebo $u_1 \neq v_1$. V prvním případě (u_0, v_0) a ve druhém případě (u_1, v_1) je pak dvojicí různých uzlů, které počítají tutéž funkci. V obou případech dostaneme spor s volbou (u, v). □

Nechť P je libovolné π-OBDD pro funkci f, jehož všechny uzly jsou dosažitelné z počátečního uzlu. Budeme konstruovat nové π-OBDD Q a částečné zobrazení $\phi : P \to Q$ tak, že

(a) Pro každý uzel v v diagramu P, je $\phi(v)$ uzel v Q, který počítá stejnou funkci jako v.

(b) Diagram Q splňuje podmínky (ii) a (iii) z Věty 1.4.

Diagram Q může být vytvářen jako sdílené OBDD. To je OBDD, které reprezentuje několik funkcí tím, že obsahuje více počátečních uzlů, jeden pro každou reprezentovanou funkci. Za OBDD pro jednu funkci pak považujeme podgraf uzlů, které jsou dosažitelné z daného počátečního uzlu. Takto definované OBDD vždy splňuje podmínku (i) z Věty 1.4 a její splnění tedy budeme předpokládat. Z vlastností (a) a (b) pak vyplývá, že část Q odpovídající dané funkci f je minimální π-OBDD pro f.

Algoritmus pro konstrukci Q a ϕ bude prohledávat P pomocí depth-first-search. Algoritmus je tedy možné použít i pro OBDD, které je reprezentované jen implicitně pomocí vhodného popisu uzlů a hran, který umožňuje jejich generování na základě jejich bezprostředních předchůdců. Tuto možnost využijeme v algoritmu pro syntézu funkcí.

Pro nalezení Q použijeme následující postup. Vrcholy P procházíme strategií depth-first-search, přičemž vytváříme dvě tabulky, které jsou na počátku prázdné. Do první tabulky ukládáme navštívené uzly v diagramu P a jim příslušné hodnoty zobrazení ϕ. Do druhé tabulky ukládáme vytvářené uzly Q.

Uzly v Q jsou reprezentovány svým popisem, který může být dvou typů. Koncový uzel je popsán svojí hodnotou 0 nebo 1. Testovací uzel je popsán trojicí: proměnná, která se testuje, odkaz na 0-následníka v Q a odkaz na 1-následníka v Q. Tato reprezentace předpokládá, že uzly vytváříme zdola nahoru, takže každý uzel vytváříme až po jeho následnících v Q.

Uzly v Q vytváříme postupně pomocí funkce find, která jako argument dostane popis požadovaného uzlu. Funkce v obou případech nejprve zjistí, zda je v Q již nějaký uzel s daným popisem obsažen. K tomu stačí pro koncové uzly evidovat, které ze dvou možných koncových uzlů již v Q je, s odkazy na tyto uzly. Pro nekoncové uzly se k nalezení případného již dříve zařazeného uzlu se stejným popisem použije tabulka všech trojic (proměnná, 0-následník, 1-následník), které jsou v Q nějakým uzlem již reprezentovány, s odkazem na příslušný uzel. Při hledání uzlu se v tabulce hledá trojice popisující uzel. Je-li trojice nalezena, je výstupem funkce find odkaz na příslušný uzel Q. Pokud takový uzel nalezen není, je vytvořen nový uzel podle daného popisu, příslušná trojice je zařazena do tabulky s odkazem na vytvořený uzel a tento odkaz je také výstupem find.

V zájmu efektivity mohou být obě popsané tabulky implementovány pomocí hashovacích funkcí a složitost jednotlivých operací s nimi lze považovat za konstantní.

Algoritmus začíná v počátečním uzlu P a prochází P pomocí depth-first-search. Pomocí první tabulky vyloučíme prohledávání následníků uzlů, které již byly dříve navštíveny. Při prvním opouštění uzlu v směrem nahoru provedeme následující akce:

- Je-li v koncový uzel s hodnotou $a \in \{0,1\}$, pak definujeme $\phi(v) =_{\text{def}} \text{find}(a)$.
- Není-li v koncový uzel, je již definováno $\phi(v_0)$ i $\phi(v_1)$, kde v_0, v_1 jsou následníci v. Je-li $\phi(v_0) = \phi(v_1)$, definujeme $\phi(v) =_{\text{def}} \phi(v_0)$. V opačném případě definujeme $\phi(v) =_{\text{def}} \text{find}(x_i, \phi(v_0), \phi(v_1))$, kde x_i je proměnná testovaná ve v.

Po provedení popsaného postupu je konstrukce Q dokončena tím, že za jeho počáteční uzel je zvolen uzel $\phi(s)$, kde s je počáteční uzel P. Lze ověřit, že pro každý uzel v v P je $\phi(v)$ uzel Q dosažitelný z $\phi(s)$. Navíc, všechny uzly Q dosažitelné z $\phi(s)$ tvoří π-OBDD pro f. Rozborem případů, ze kterých se algoritmus skládá, a s využitím definice funkce find lze ověřit, že jsou zaručeny podmínky (a) a (b). Z Věty 1.4 tedy plyne následující.

Věta 1.5 *Zkonstruovaný diagram Q je minimální π-OBDD pro funkci f reprezentovanou vstupním diagramem P. Počet operací vkládání a hledání uzlů v tabulkách je lineární v počtu uzlů P.*

Pro nalezené minimální OBDD lze řešit problém splnitelnosti reprezentované funkce f tak, že zjistíme, zda minimální reprezentace f obsahuje koncový uzel s hodnotou 1. Test splnitelnosti ale nevyžaduje minimální OBDD, protože pro libovolné OBDD je splnitelnost ekvivalentní dosažitelnosti výstupního uzlu 1 z počátečního uzlu.

Testování ekvivalence funkcí f a g lze díky Větě 1.2 převést na testování izomorfismu minimálních reprezentací. Problém izomorfismu je v tomto případě snadno řešitelný, protože se jedná o labelované struktury. Test izomorfie není dokonce nutný, jestliže OBDD pro f a g jsou reprezentovány ve sdíleném OBDD. V tomto případě jsou funkce f a g totožné právě tehdy, když jsou reprezentovány stejným počátečním uzlem.

Věta 1.2 dává uzly minimálního OBDD do vzájemně jednoznačné korespondence s množinou subfunkcí reprezentované funkce. Uzly minimálního sdíleného OBDD pro k-tici funkcí jsou v korespondenci se subfunkcemi ve sjednocení odpovídajících množin subfunkcí všech funkcí reprezentované k-tice. Pokud se tyto množiny překrývají, může nastat úspora v počtu uzlů.

1.2 Syntéza OBDD

Syntézou se rozumí kombinování funkcí pomocí Booleovských spojek. Popíšeme syntézu pro obecnou k-ární spojku $\alpha : \{0,1\}^k \to \{0,1\}$. Nechť π je libovolné uspořádání proměnných.

Nechť $f_1, f_2, \ldots, f_k$ jsou funkce proměnných $x_1, \ldots, x_n$ a nechť f_i je reprezentována π-OBDD P_i pro $i = 1, 2, \ldots, k$. Definujme π-OBDD P na množině uzlů $P_1 \times P_2 \times \ldots \times P_k$ následovně.

- Počáteční uzel P je $\langle u_1, u_2, \ldots, u_k \rangle$, kde u_i je počáteční uzel P_i.
- Nechť $\langle v_1, v_2, \ldots, v_k \rangle$ je uzel P a nechť I je množina indexů i takových, že v_i je koncový uzel P_i. Uvažme funkci, která vznikne dosazením hodnoty uzlu v_i za y_i pro $i \in I$ do funkce $\alpha(y_1, y_2, \ldots, y_k)$. Nechť J je množina indexů proměnných y_i, na kterých takto získaná funkce závisí. Pokud $J = \emptyset$, je získaná funkce konstantní a nechť c je její hodnota.
  - Jestliže $J = \emptyset$, pak $\langle v_1, v_2, \ldots, v_k \rangle$ je koncový uzel s hodnotou c.
  - Jestliže $J \neq \emptyset$, pak uzel $v = \langle v_1, v_2, \ldots, v_k \rangle$ testuje proměnnou $x(v)$, která je určena jako proměnná s nejmenším indexem v π mezi proměnnými $x(v_i)$ pro $i \in J$. Následník v se dostane tak, že v souřadnicích, kde $x(v) = x(v_i)$, přejdeme na příslušného následníka v_i, a v ostatních souřadnicích uzel v_i zachováme.

Věta 1.6 *Zkonstruované OBDD počítá funkci $\alpha(f_1, f_2, \ldots, f_k)$.*

Důkaz. K důkazu pouze poznamenáme, že výpočet vytvořeného OBDD přesně kopíruje paralelní výpočet podle všech OBDD P_i pro $i = 1, 2, \ldots, k$, kde jsou v každém kroku zastaveny výpočty pro ty P_i, které čekají na čtení proměnné nebo jejichž hodnota již není k vyčíslení výsledku zapotřebí. Výpočet je ukončen, jakmile máme dostatečnou informaci k vyhodnocení výsledku. □

Poměrně komplikovaná definice koncového uzlu v P se v konkrétních případech může podstatně zjednodušit. Například, je-li $k = 2$ a $\alpha(y_1, y_2) = y_1 y_2$, můžeme množinu J definovat jako doplněk I, protože nemůžeme fixací y_1 nebo y_2 dostat funkci, která není konstantní, ale závisí jen na některých zbylých proměnných. V praktických implementacích se někdy jako základní operace používá operace $\mathrm{ite}(x, y, z)$ (if-then-else), která je definována vztahy $\mathrm{ite}(0, y, z) = y$ a $\mathrm{ite}(1, y, z) = z$. Pomocí této operace a negace pak lze realizovat všechny binární spojky, protože stačí testovat jednu z jejích proměnných a pak realizovat příslušnou funkci jedné proměnné. Například $x \wedge y = (x, 0, y)$, $x \vee y = (x, y, 1)$ a $x \oplus y = (x, y, \neg y)$.

Zkonstruované součinové OBDD může být velmi vzdálené od minimálního. Pro nalezení minimálního OBDD se postupuje následovně. Součinové OBDD není konstruováno celé, ale pracujeme s reprezentací jeho uzlů pomocí k-tic uzlů diagramů $P_1, \ldots, P_k$. Tuto reprezentaci použijeme v algoritmu minimalizace z minulé podkapitoly. Tím dostaneme následující větu.

Věta 1.7 *Minimální π-OBDD pro funkci $\alpha(f_1, \ldots, f_k)$, kde f_i je funkce reprezentovaná π-OBDD P_i, lze nalézt v čase $O(|P_1| \cdot \ldots \cdot |P_k|)$, pokud složitost operací vkládání a hledání v tabulkách považujeme za konstantní.*

Popsaný algoritmus konstruuje jen ty uzly součinového diagramu, které jsou dosažitelné z počátečního uzlu přes uzly, které ještě nesplňují podmínku ukončení. Tato úspora je jen heuristická. Složitost nejhoršího případu zůstává rovna součinu velikosti vstupních diagramů. Otázka, zda lze výsledné π-OBDD zkonstruovat v čase polynomiálním od jeho velikosti (od velikosti výstupu), je otevřený problém.

1.3 Verifikace obvodů

Smyslem verifikace obvodů pomocí OBDD je provést úplný test korektnosti obvodu pro všechny kombinace vstupních hodnot. Pro každé hradlo obvodu uvažujeme funkci, kterou dané hradlo počítá. Vstupní uzly obvodu počítají projekce (proměnné) a jejich negace. Tyto funkce lze vyjádřit pomocí OBDD s jedním testovacím uzlem. Funkce počítané v dalších hradlech jsou vázány rekurentními vztahy vyplývajícími ze struktury obvodu. Pro funkci, kterou počítá hradlo $\alpha(g_1, g_2, \ldots, g_k)$, kde $g_1, g_2, \ldots, g_k$ jsou jeho předchůdci, nalezneme OBDD pomocí algoritmu syntézy. Ve všech krocích pracujeme s minimálními OBDD reprezentovanými ve sdíleném OBDD.

Popsaným postupem nalezneme OBDD pro funkci reprezentovanou ve výstupním hradle obvodu. Tuto funkci pak porovnáme s reprezentací funkce, která má být obvodem počítána. Toto referenční OBDD můžeme získat různými způsoby, například analýzou již dříve verifikovaného obvodu pro danou funkci nebo přímou konstrukcí podle definice funkce.

Efektivita postupu podstatně závisí na zvoleném uspořádání π. Volba optimálního uspořádání je NP-úplný problém, proto se k jeho řešení používají heuristiky. Jmenujme v této souvislosti postup nazývaný sifting, viz. Ingo Wegener, Branching Programs and Binary Decision Diagrams: Theory and Applications (Monographs on Discrete Mathematics and Applications), 2000. Sifting je heuristika, která hledá lokálně optimální uspořádání tím, že se zvolí některá proměnná a vyzkouší se všechny její možné pozice v uspořádání a vybere se pozice, která vede na nejmenší OBDD. Tento postup se opakuje postupně se všemi proměnnými.

Pro manipuaci s OBDD existují dostupné balíky programů, například CUDD na adrese http://vlsi.colorado.edu/~fabio/CUDD.

2 Rozhodovací stromy pro funkce definované současně DNF a CNF

Velikost CNF pro f je rovna velikosti DNF pro $\neg f$. Místo současné reprezentace pomocí DNF a CNF tedy můžeme uvažovat DNF pro f a $\neg f$. První odhad ukazuje souvislost délky monomů v DNF pro f a $\neg f$ a hloubky rozhodovacího stromu pro f.

Věta 2.1 *Jestliže funkce f je reprezentovaná DNF s monomy délky nejvýše s a $\neg f$ je reprezentovaná DNF s monomy délky nejvýše t, pak f je reprezentována stromem hloubky nejvýše st.*

Důkaz. Nejprve si uvědomme, že každý monom z DNF pro f a každý monom z DNF pro $\neg f$ obsahují alespoň jednu dvojici komplementárních literálů. Kdyby ne, bylo by možné oba tyto monomy splnit stejným ohodnocením, což by byl spor s tím, že uvažujeme reprezentace funkcí f a $\neg f$.

Tvrzení dokážeme indukcí podle součinu st. Pokud je $st = 0$, pak je funkce konstantní a lze ji vyjádřit stromem hloubky 0, který obsahuje pouze kořen.

Pokud $st > 0$, zvolíme libovolný monom z DNF pro f a vytvoříme strom τ, který testuje právě všechny proměnné v něm. V listech tohoto stromu ještě nemusí být známa hodnota funkce f a bude potřeba připojit vhodné podstromy. Jestliže u je některý z listů τ, pak je k němu potřeba připojit podstrom, který počítá restrikci funkce f, kde jsou zafixovány proměnné testované na cestě z kořene τ do u. Tato restrikce funkce f je vyjádřitelná pomocí DNF s monomy délky nejvýše s, protože stačí použít restrikci DNF pro f. Negace této restrikce funkce f je vyjádřitelná pomocí DNF s monomy délky nejvýše $t-1$, protože každý monom výchozí DNF pro $\neg f$ obsahuje alespoň jednu proměnnou, která je zafixována.

Protože $s(t-1) < st$, můžeme použít indukční předpoklad a dostaneme, že funkce, které je třeba počítat v listech stromu τ, lze vyjádřit pomocí stromů hloubky nejvýše $s(t-1)$. Protože τ má hloubku nejvýše s, dostaneme hloubku výsledného stromu nejvýše st, jak bylo požadováno. □

Lemma 2.2 *Jestliže ϕ je tautologie složitosti l, pak v ní existuje monom délky nejvýše* $\log_2 l$.

Důkaz. Uvažme náhodné ohodnocení proměnných, ve kterém jsou hodnoty proměnných nezávislé a mají rovnoměrné rozdělení na $\{0,1\}$. Monom délky k je splněn s pravděpodobností $1/2^k$. Protože je ϕ splněna pro každé ohodnocení proměnných, je některý z l monomů splněn s pravděpodobností alespoň $1/l$. Pro tento monom musí být splněno $1/2^k \geq 1/l$, tedy jeho délka je $k \leq \log_2 l$. □

Následující věta ukazuje souvislost složitosti DNF pro f a $\neg f$ a velikosti rozhodovacího stromu pro f.

Věta 2.3 *Označme $s = dnf(f)$, $t = dnf(\neg f)$, $k = \lfloor log_2(s+t)ln(st) \rfloor + 1$. Pak $dt(f) \leq \sum_{i=0}^{k} \binom{n}{i}$.*

Důkaz. Rozhodovací strom budeme konstruovat tak, že každému uzlu přiřadíme dvojici DNF (ϕ_1, ϕ_2), kde ϕ_1 vyjadřuje funkci, kterou má počítat daný uzel, a ϕ_2 je reprezentace její negace. Nejprve vytvoříme kořen stromu a přiřadíme mu dvojici DNF pro f a $\neg f$, které mají složitost s a t. Dalším uzlům budou přiřazeny restrikce těchto formulí podle hodnot testovaných proměnných, což jsou opět dvojice formulí, které jsou navzájem negací. V každém uzlu v stromu budeme volit testovanou proměnnou tak, aby alespoň jeden podstrom počítal funkci v určitém smyslu jednodušší než funkce počítaná ve v. Jako míru složitosti použijeme $s't'$, kde dvojice čísel (s', t') jsou složitosti jednotlivých DNF ve dvojici (ϕ_1, ϕ_2) přiřazených danému uzlu.

Nechť uzlu v jsou přiřazeny DNF (ϕ_1, ϕ_2) se složitostmi (s', t'). Protože disjunkce $\phi_1 \vee \phi_2$ je tautologie, obsahuje podle Lemmatu 2.2 monom m délky $r \leq \log_2(s'+t')$. Předpokládejme nejprve, že m je v části ϕ_1. Každý monom ϕ_2 obsahuje literál, který je opačný k nějakému literálu m. Monom m tedy obsahuje literál, který je ve sporu s alespoň t'/r monomy ϕ_2. Vyberme některý takový literál a použijme jej pro test v uzlu v. Podstrom, který dostaneme pro nesplnění tohoto literálu, bude ohodnocen dvojicí DNF se složitostmi nejvýše $(s'-1, t')$. Podstrom, který dostaneme pro splnění vybraného literálu, bude ohodnocen dvojicí DNF se složitostmi nejvýše $(s', t'(1-1/r))$. Hranu, která odpovídá tomuto druhému případu, tedy splnění vybraného literálu, budeme nazývat zjednodušující hranou. Tato hrana vede do uzlu, ve kterém je součin délek přiřazených DNF nejvýše $s't'(1-1/r)$. Pokud je vybraný literál m v části ϕ_2, dostaneme podobnou úvahou, že při jeho splnění dostaneme dvojici DNF se složitostmi nejvýše $(s'(1-1/r), t')$. Odpovídající hranu ve stromu budeme opět nazývat zjednodušující a součinová míra složitosti v uzlu, do kterého vede tato hrana, je opět nejvýše $s't'(1-1/r)$.

Pro odhad počtu uzlů ve stromu budeme cestu z kořene do libovolného uzlu zapisovat jako posloupnost znaků 0 a 1 délky nejvýše n, kde 1 označuje přechod po zjednodušující hraně a 0 po druhé z hran v daném uzlu. Uvažme uzel, do kterého vede cesta, jejíž zápis obsahuje i jedniček. V nerovnostech pro všechny zjednodušující hrany platí $r \leq \log_2(s'+t') \leq \log_2(s+t)$, a tedy v uvažovaném uzlu bude součinová míra splňovat

$$s't' \leq st\left(1 - \frac{1}{\log_2(s+t)}\right)^i \leq st\,\mathrm{e}^{-\frac{i}{\log_2(s+t)}} .$$

Pokud uzel není list stromu, platí $1 \leq s't'$, a tedy

$$0 \leq \ln(st) - \frac{i}{\log_2(s+t)} ,$$

což po úpravě dává

$$i \leq \ln(st) \log_2(s+t) \ .$$

Pro vnitřní uzly stromu tedy platí $i < k$ a pro všechny uzly pak $i \leq k$. Posloupnosti přiřazené listům stromu doplníme znaky 0 na délku n. Různé listy budou i po tomto doplnění reprezentovány různými posloupnostmi. Protože tyto posloupnosti obsahují nejvýše k znaků 1, je počet listů ve stromu nejvýše

$$\sum_{i=0}^{k} \binom{n}{i} \ ,$$

čímž je tvrzení dokázáno. □

Pro $k \leq n/2$ platí následující nerovnosti

$$\sum_{i=0}^{k} \binom{n}{i} \leq 2^{H(k/n)\, n} \leq \left(\frac{ne}{k}\right)^k \ .$$

Pomocí tohoto odhadu a Věty 2.3 lze dokázat (i) v následující větě. Bod (ii) uvádíme bez důkazu.

Věta 2.4 *Označme $N(f) = dnf(f) + dnf(\neg f)$. Pak platí následující.*
(i) Pro libovolnou f platí $dt(f) \leq 2^{O(\log n \log^2 N(f))}$, kde n je počet proměnných funkce f.
(ii) Existuje nekonečně mnoho funkcí f, pro které platí $dt(f) \geq 2^{\Omega(\log^2 N(f))}$.

3 Operace s rozhodovacími stromy

Jestliže máme reprezentace funkcí $f_1, f_2, \ldots, f_k$ pomocí rozhodovacích stromů a $\alpha : \{0,1\}^k \to \{0,1\}$ je obecná k-ární spojka, pak reprezentaci $\alpha(f_1, f_2, \ldots, f_k)$ získáme v čase lineárním v součinu velikostí vstupních stromů následovně. Ve stromu pro f_1 připojíme ke každému listu strom pro f_2. Pokud $k > 2$, pokračujeme přidáváním stromů pro f_i, $i = 3, \ldots, k$ ke každému listu stromu z předchozího kroku. Velikost výsledného stromu je součin velikosti stromů pro $f_1, f_2, \ldots, f_k$. Každému listu spojeného stromu odpovídá jednoznačně určená cesta z kořene, která postupně projde listy stromů pro f_i pro všechna $i = 1, \ldots, k$. Ohodnocení listů stromů pro f_i určuje hodnoty těchto funkcí pro všechna $i = 1, \ldots, k$. Dosazením těchto hodnot do α dostaneme ohodnocení uvažovaného listu spojeného stromu.

Konstrukcí z předchozího odstavce získáme strom, který počítá $\alpha(f_1, f_2, \ldots, f_k)$, ale může obsahovat zbytečné testy, protože větve tohoto spojeného stromu mohou testovat některé proměnné opakovaně. V takovém případě je možné strom zmenšit tím, že pro každý jeho uzel zjistíme, zda v něm testovaná proměnná byla již testována na cestě z kořene do daného uzlu. Pokud ano, pak uzel odstraníme

a nahradíme tím jeho následníkem, který odpovídá hodnotě testované proměnné konzistentní s předchozím testem této proměnné.

Test ekvivalence pro stromy, tedy test rovnosti dvou funkcí f a g, které jsou reprezentovány pomocí stromů, lze provést tak, že sestrojíme strom pro $f \oplus g$. Tento strom reprezentuje nulovou funkci právě tehdy, když $f = g$.

Alternativně můžeme projít všechny dvojice listů ze stromu pro f a pro g, které dávají opačné hodnoty, a pro každou takovou dvojici otestovat, zda cesty, které do nich vedou z kořene příslušného stromu, obsahují komplementární literály. Pokud každá dvojice uvažovaných cest obsahuje dvojici komplementárních literálů, je $f = g$. Pokud najdeme dvojici listů s opačnými hodnotami, do kterých vedou cesty, které neobsahují komplementární literály, pak existuje ohodnocení, které je konzistentní s oběma těmito cestami, a funkce f a g tedy na něm mají různé hodnoty. V tomto případě je $f \neq g$.

Věta 3.1 *Minimální rozhodovací strom pro funkci n proměnných lze najít na RAM v prostoru $O(3^n)$ a čase $poly(n)3^n$.*

Důkaz. Budeme hledat minimální strom pro restrikce dané funkce postupně na všechny podkrychle počínaje jednobodovými podkrychlemi. Pro každou podkrychli je třeba uložit volbu proměnné, která bude testována v kořeni stromu a velikost odpovídajícího minimálního stromu. Pro konstrukci minimálního stromu na podkrychli dimenze d při znalosti minimálních stromů pro všechny podkrychle dimenze $d-1$ stačí projít d možných proměnných pro test v kořeni a vybrat tu, která vede na strom, jehož levý a pravý podstrom dávají v součtu minimální složitost. Počet všech podkrychlí je 3^n, což dokazuje požadované odhady složitosti. □

4 Neuniformní Turingův stroj

Posloupnost B.f. lze přirozeným způsobem chápat jako jazyk a lze uvažovat o jeho reprezentaci Turingovým strojem. Při této reprezentaci máme pro všechny funkce v posloupnosti stejný algoritmus výpočtu. Při reprezentaci pomocí Booleovských modelů máme možnost zvolit pro každou funkci v posloupnosti její reprezentaci nezávisle na ostatních funkcích. To zvyšuje sílu těchto modelů, protože mohou například reprezentovat nerekurzivní posloupnosti funkcí.

Uvedené dva typy reprezentace se rozlišují pomocí pojmů uniformní a neuniformní model. *Uniformní model* je takový, že pro všechny délky vstupu použije stejný algoritmus. Mezi uniformní třídy patří základní třídy jako LOG $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE. *Neuniformní model* může pro každou délku vstupu použít jiný algoritmus. Všechny modely pro reprezentaci Booleovských funkcí zkoumané v předchozích sekcích tohoto textu patří mezi neuniformní modely.

Turingův stroj je základním reprezentantem uniformních modelů, ale existuje možnost, jak pomocí TS studovat i neuniformní modely pomocí tzv. advice funkcí. Protože takto vzniklý neuniformní TS umožňuje snadněji formulovat vztah mezi uniformní a neuniformní složitostí, popíšeme tento model podrobněji.

Advice funkce je libovolná funkce $\alpha : \mathrm{N} \to \{0,1\}^*$. Turingův stroj s advice 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 advice funkci α budeme jako T/α značit třídu jazyků, pro které existuje Turingův stroj s advice 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.

V následující větě porovnáváme třídy jazyků a třídy posloupností Booleovských funkcí. Pro tento účel budeme posloupnost Booleovských funkcí, v níž je f_i funkcí n_i proměnných, považovat za reprezentaci jazyka

$$\{w \in \{0,1\}^*;\ \text{existuje } i \text{ tak, že } n_i = |w| \text{ a } f_i(w) = 1\}.$$

Věta 4.1 *Platí následující rovnosti tříd:*

Poly(rozhodovací diagramy)	=	LOG/poly,
Poly(obvody)	=	P/poly,
Poly(sekvenční obvody)	=	PSPACE/poly.

Důkaz. Uvedeme jen stručný důkaz této věty. Inkluze $\subseteq$ dostaneme ve všech případech tím, že jako $\alpha(n_i)$ zvolíme některou reprezentaci funkce f_i velikosti polynomiální v n_i v daném modelu. Pro n, které je různé od všech n_i, zvolíme jako $\alpha(n)$ vždy tutéž reprezentaci nulové funkce. Turingův stroj v pravé straně dokazované inkluze má při vstupu $w \in \{0,1\}^n$ k dispozici reprezentaci $\alpha(n)$ a provede simulaci jejího výpočtu na vstup w a vydá získaný výsledek.

Inkluzi $\supseteq$ dokážeme v prvním případě následovně. Vezmeme TS v LOG/poly, který přijímá jazyk $L \subseteq \{0,1\}^*$, a zvolme libovolné přirozené číslo n. Popíšeme konstrukci rozhodovacího diagramu polynomiální velikosti, který dává pro vstup $w \in \{0,1\}^n$ hodnotu 1 právě tehdy, když $w \in L$.

TS pro L doplníme čítačem počtu provedených kroků. K tomu stačí dodatečný prostor logaritmické velikosti. Za konfiguraci získaného stroje budeme považovat vnitřní stav jeho jednotky, obsah pracovní pásky, polohu čtecí hlavy na vstupní pásce (která je pouze pro čtení), na pracovní pásce a na pásce s $\alpha(n)$. Z předpokladů plyne, že existuje pouze polynomiálně mnoho takovýchto konfigurací. Dále, při fixovaném $\alpha(n)$ závisí přechody z těchto konfigurací do jejich následníků jen na obsahu pole vstupní pásky, kde je hlava.

Vytvoříme graf, kde každé z těchto konfigurací bude odpovídat jeden uzel. Index pole vstupní pásky, na jehož obsahu závisí, do které konfigurace výpočet přejde, je součástí konfigurace. V každém uzlu diagramu je tedy určeno, která pozice vstupního slova rozhoduje, do které konfigurace přejde další krok výpočtu. Každý uzel grafu konfigurací tedy lze považovat za uzel rozhodovacího diagramu, který testuje určitou vstupní proměnnou (pozici slova w). Protože konfigurace obsahují informaci o počtu dosud provedených kroků, je získaný diagram acyklický. Konfigurace odpovídající koncovým stavům TS jsou výstupní uzly diagramu. Těmto uzlům přiřadíme hodnotu 0 nebo 1 podle toho, zda odpovídají zamítajícímu nebo přijímajícímu stavu TS. Tím jsme získali rozhodovací diagram, který jsme požadovali.

Inkluzi $\supseteq$ v dalších dvou případech dokážeme následovně. Vezměme libovolný TS v P/poly, resp. v PSPACE/poly, který přijímá nějaký jazyk $L \subseteq \{0,1\}^*$, a zvolme libovolné přirozené číslo n. Zkonstruujeme kombinační, resp. sekvenční obvod velikosti polynomiální v n, který vydá pro vstup w délky n hodnotu 1 právě tehdy, když $w \in L$.

V obou případech je pracovní prostor omezen polynomem, který označíme $p(n)$. Můžeme tedy omezit délku pracovních pásek na $p(n)$ beze změny funkce TS. Stav řídící jednotky a znaky na pracovních páskách vyjádříme pomocí posloupností bitů pevné délky. Ke každému poli každé pásky ještě přidáme bit, který bude signalizovat přítomnost čtecí hlavy na daném poli. Kód stavu řídící jednotky, kódy všech znaků na páskách a bity pro polohu hlavy zřetězíme v pevně zvoleném pořadí do jedné posloupnosti bitů, kterou budeme nazývat konfigurací stroje. Pomocí definice TS a tabulky jeho řídící jednotky lze sestrojit obvod, který dostane jako vstup konfiguraci stroje a jako výstup vydá konfiguraci následující, tj. po provedení jednoho kroku výpočtu TS. Pokud je TS v koncovém stavu, bude vypočtená konfigurace kopií vstupní konfigurace.

V případě TS z P/poly vezmeme polynomiální odhad času $t(n)$ a vytvoříme posloupnost $t(n)$ kopií výše popsaného obvodu. První kopie obvodu dostane na vstup počáteční konfiguraci se slovem w. Bity popisující slovo w budou vstupní proměnné obvodu, ostatní bity počáteční konfigurace budou konstanty. Výstup každé kopie obvodu kromě poslední je připojen na vstup následující kopie. Výstup

poslední kopie obvodu je připojen na vstup dalšího obvodu, který otestuje, zda se simulovaný TS dostal do přijímajícího stavu nebo ne. Popsaným spojením obvodů jsme získali obvod velikosti $O(t(n)p(n))$, který rozhoduje příslušnost w do L. Posloupnost těchto obvodů pro všechna n dokazuje požadovanou inkluzi.

Pro TS z PSPACE/poly vezmeme jednu kopii výše popsaného obvodu pro jeden krok a její výstup připojíme opět na jeho vstup pomocí zpětné vazby kontrolované hodinovými impulzy. V krocích určených těmito impulzy bude tento obvod simulovat výchozí TS. K výstupu obvodu bude ještě připojen obvod, který bude testovat, zda již TS došel do koncového stavu nebo ne. Tím získáme sekvenční obvod polynomiální velikosti, který rozhoduje příslušnost $w \in \{0,1\}^n$ do L. Stejně jako v předchozím případě, posloupnost těchto obvodů pro všechna n dokazuje požadovanou inkluzi. □

5 Karp-Liptonova věta

Nejprve popíšeme třídy, které tvoří polynomiální hierarchii. K tomu použijeme kvantifikátory $\exists^m$ a $\forall^m$, kde horní index m u kvantifikátoru znamená maximální délku slov, která tvoří rozsah kvantifikátoru. Polynomiální hierarchie splňuje následující.

Věta 5.1 *Jazyk L patří*
(i) do Σ_k právě tehdy, když existují polynomy $p_1, \ldots, p_k$ a relace $R \in \mathrm{P}$ tak, že pro každé slovo w platí

$$w \in L \iff (\exists^{p_1(|w|)} x_1)(\forall^{p_2(|w|)} x_2) \ldots (Q^{p_k(|w|)} x_k) R(w, x_1, x_2, \ldots, x_k),$$

kde Q je $\exists$, pokud k je liché a $\forall$, pokud k je sudé.
(ii) do Π_k právě tehdy, když existují polynomy $p_1, \ldots, p_k$ a relace $R \in \mathrm{P}$ tak, že pro každé slovo w platí

$$w \in L \iff (\forall^{p_1(|w|)} x_1)(\exists^{p_2(|w|)} x_2) \ldots (Q^{p_k(|w|)} x_k) R(w, x_1, x_2, \ldots, x_k),$$

kde Q je $\forall$, pokud k je liché a $\exists$, pokud k je sudé.

Věta předpokládá, že uvažujeme polynomy, které lze snadno vyčíslit, například polynomy s celočíselnými nebo racionálními koeficienty. Pokud bychom uvažovali polynomy, jejichž koeficienty jsou reálná čísla s libovolně složitým popisem, pak věta neplatí.

Jazyk S nad abecedou Γ nazveme self-reducibilní, pokud existuje funkce f definovaná na Γ^* taková, že pro každé $w \in \Gamma^*$ je $f(w)$ buď prvek $\{0,1\}$ a v tom případě platí

$$w \in S \iff f(w) = 1$$

nebo $(w_1, \ldots, w_l)$, kde všechna w_i, $i = 1, \ldots, l$ mají délku menší než w a platí

$$w \in S \iff w_1 \in L \vee \ldots \vee w_l \in L \ .$$

Problém SAT je self-reducibilní. Pro formuli ϕ bez proměnných definujeme $f(\phi)$ jako hodnotu formule ϕ. Pro formuli ϕ, která obsahuje alespoň jednu proměnnou, definujeme $f(\phi)$ jako (ϕ_0, ϕ_1), kde ϕ_1 a ϕ_2 jsou formule vzniklé z ϕ dosazením 0 a 1 za první proměnnou formule ϕ. Předpokládáme, že kódování formulí je takové, že dosazením konstanty za proměnnou se zápis formule vždy zkrátí.

Věta 5.2 (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 instance velikosti nejvýše n. Předpokládejme dále, že $L \in \Pi_2$. Pak existuje relace $R \in \mathrm{P}$, taková, že pro vhodné polynomy p_1, p_2 platí

$$w \in L \iff (\forall^{p_1(|w|)} x_1)(\exists^{p_2(|w|)} x_2) R(w, x_1, x_2) \ .$$

Predikát $(\exists^{p_2(|w|)} x_2) R(w, x_1, x_2)$ je v NP, tedy existuje poly-vyčíslitelná funkce g, která pro $|x_1| \leq p_1(|w|)$ splňuje $(\exists^{p_2(|w|)} x_2) R(w, x_1, x_2) \iff g(w, x_1) \in S$. Platí tedy

$$w \in L \iff (\forall^{p_1(|w|)} x_1) g(w, x_1) \in S \ .$$

Nechť $p_3(|w|)$ je polynomiální horní mez na délku $g(w, x_1)$ při $|x_1| \leq p_1(|w|)$. Pokud C je obvod rozpoznávající slova v S délky nejvýše $p_3(|w|)$, pak platí také

$$w \in L \iff (\forall^{p_1(|w|)} x_1) C(g(w, x_1)) \ . \qquad (1)$$

Naším cílem je sestrojit Σ_2 formuli s argumentem w, která neobsahuje popis konkrétního obvodu C pro S a umožňuje vyjádřit $w \in L$ jen na základě existence takového obvodu C. Požadovaná formule bude utvořena následovně. Její existenční kvantifikátor bude kvantifikovat přes všechny obvody C velikosti nejvýše $s(p_3(|w|))$. Zbytek formule se bude skládat z konjunkce dvou podformulí. První zkontroluje, že uhádnutý obvod skutečně řeší S pro instance délky nejvýše $p_3(|w|)$ pomocí self-reducibility S a druhá využije $C(g(w, x_1))$ ke zjištění, zda $g(w, x_1) \in S$. Formule s volnou proměnnou w má tvar

$$(\exists^{s(p_3(|w|))} C) \left[(\forall^{p_3(|w|)} u) \left[C(u) \equiv C(f(u)) \right] \wedge (\forall^{p_1(|w|)} x_1) C(g(w, x_1)) \right] , \qquad (2)$$

kde $C(u) \equiv C(f(u))$ je zkratka za test, který pro $f(u) \in \{0,1\}$ ověří $C(u) \equiv f(u)$ a pro $f(u) = (u_1, \ldots, u_l)$ ověří $C(u) \equiv C(u_1) \vee \ldots \vee C(u_l)$. Formule (2) je ekvivalentní pravé straně formule (1) pro libovolný korektní obvod C pro S a vyjadřuje tedy $w \in L$. Tím je dokázáno, že $L \in \Sigma_2$. □

Důsledek 5.3 $\mathrm{SAT} \in \mathrm{P}/poly \Rightarrow \Pi_2 = \Sigma_2$

Protože polynomiální převod SAT na libovolnou NP-úplnou úlohu umožňuje převést polynomiální obvody pro tuto úlohu na polynomiální obvody pro SAT, platí důsledek obecně pro jakoukoli NP-úplnou úlohu místo SAT.

Později dokážeme, že BPP⊆P/poly. Spolu s Důsledkem 5.3 to implikuje následující.

Důsledek 5.4 $\mathrm{SAT} \in \mathrm{BPP} \Rightarrow \Pi_2 = \Sigma_2$

Stejně jako v předchozím případě dostaneme, že Důsledek 5.4 platí pro libovolnou NP-úplnou úlohu místo SAT. Pokud je polynomiální hierarchie nekonečná, pak žádná NP-úplná úloha není v BPP. Kdyby totiž patřil nějaký NP-úplný jazyk do BPP, pak by každý jazyk z NP patřil do BPP, tedy i SAT. Z toho podle Důsledku 5.4 plyne $\Pi_2 = \Sigma_2$ a polynomiální hierarchie by byla rovna své druhé hladině.

Poznamenejme, že důsledek 5.4 má význam pouze za předpokladu, že neplatí nebo neumíme dokázat, že P=BPP. Pokud bychom předpokládali, že P=BPP, pak je $\mathrm{SAT} \in \mathrm{BPP}$ právě tehdy, když $\mathrm{SAT} \in \mathrm{P}$, a následující triviální tvrzení

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

zesiluje Důsledek 5.4.

6 Počet splňujících ohodnocení pro read-once rozhodovací diagramy

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.

Pro výpočet počtu splňujících ohodnocení přiřaďme každému uzlu u diagramu počet ohodnocení $q(u)$, pro které výpočet projde uzlem u. Pro počáteční uzel u je $q(u) = 2^n$. Předpokládejme, že u není počáteční, jeho předchůdci jsou právě uzly $\{v_1, \ldots, v_k\}$ a z každého v_j vede do u právě jedna hrana. Dokažme, že pak platí

$$q(u) = \frac{1}{2} \sum_{j=1}^{k} q(v_j) \ . \tag{3}$$

Zvolme některý z uzlů v_j. Protože diagram je read-once, proměnná, která je ve v_j testována, nebyla testována na žádné cestě z počátečního uzlu do v_j. Množina ohodnocení, které došly do v_j, se tedy skládá z dvojic vstupů, které se liší jen v hodnotě této proměnné. Po hraně z v_j do u tedy projde právě $q(v_j)/2$ ohodnocení. Z toho už plyne (3).

Počet splňujících ohodnocení reprezentované funkce je roven $q(u^+)$, kde u^+ označuje přijímající uzel.

Výpočet $q(u)$ pro všechny uzly u v diagramu lze provést průchodem do šířky shora nebo průchodem do "hloubky" zdola. Při druhém z uvedených postupů zahájíme výpočet $q(u)$ nejprve pro $u = u^+$. Pro dokončení výpočtu pro libovolný uzel u je potřeba nejprve provést výpočet pro všechny jeho předchůdce, po které ještě proveden není. To vede na postupné procházení grafu zdola.

7 Pravděpodobnostní test neekvivalence pro read-once rozhodovací diagramy

K tomu, abychom popsali pravděpodobnostní test ekvivalence, 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

$$f^P_v = (1 - x_i) f^P_{v_0} + x_i f^P_{v_1} \ . \tag{4}$$

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 7.1 *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 $f_v(x) = f_v^P(x)$, 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 $f_v^P(x)$ jsou stejné konstanty. Pro další uzly dokážeme identitu indukcí pomocí vztahu (4). □

Polynom $f_v^P(x)$ 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 7.2 *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 (4). □

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

Lemma 7.3 *Polynom f^P je multilineární polynom.*

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

Lemma 7.4 *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 7.5 *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 7.4 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 7.5 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 7.6 *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 7.6 umožňuje alternativní důkaz Lemmatu 7.4, když použijeme $M = \{0,1\}$. Kdyby existovalo ohodnocení $a \in T^n$, pro které je $h(a) \neq 0$, pak podle

Lemmatu 7.6 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 7.7 *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 $g(a'_1, \ldots, a'_n) \neq 0$. □

Pro interpretaci polynomu f^P může být užitečné následující tvrzení.

Věta 7.8 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 7.9 *Nechť jsou hodnoty $x_i \in \{0,1\}$ voleny náhodně, nezávisle a tak, že $\mathrm{P}(x_i = 1) = p_i$. Pak $P(f(x_1, \ldots, x_n) = 1) = f^p(p_1, \ldots, p_n)$.*

Důkaz. Tvrzení plyne z Věty 7.8 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 (4). □

Lemma 7.9 dává alternativní způsob výpočtu počtu 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.

Jednoduchý 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 7.7 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 7.10 *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 7.6 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.

Opakovaný test neekvivalence.
Vstup jsou dva diagramy P_1 a P_2, množina M, $|M|=2n$ a parametr k.
Opakuj k krát výše popsaný jednoduchý test neekvivalence
s využitím nezávisle generovaných náhodných bitů.
Jestliže alespoň jedno z těchto opakování nalezne důkaz neekvivalence,
pak tento důkaz je výstupem algoritmu, jinak je výstupem $\emptyset$.

Uvedený postup potřebuje $k(\log_2 n+1)n$ náhodných bitů.

Věta 7.11 *Nechť P_1 a P_2 jsou dva read-once rozhodovací diagramy s n proměnnými a k přirozené číslo. Pak pro výše popsaný test neekvivalence se vstupem P_1, P_2, k platí následující.*
(i) Jsou-li P_1 a P_2 ekvivalentní, pak vždy vydá odpověď $\emptyset$.
(ii) Pokud jsou neekvivalentní, pak pravděpodobnost, že algoritmus nalezne důkaz neekvivalence P_1 a P_2, je alespoň $1-\frac{1}{2^k}$.

Důkaz. Pro ekvivalentní diagramy vydá jednoduchý test ekvivalence vždy hodnotu $\emptyset$ a tuto hodnotu tedy vydá i celý test neekvivalence. Při jednom opakování jednoduchého testu neekvivalence je pravděpodobnost výsledku $\emptyset$ pro neekvivalentní diagramy nejvýše $\frac{1}{2}$. Při k opakováních je tedy tato pravděpodobnost nejvýše $\frac{1}{2^k}$. Pro neekvivalentní diagramy je tedy pravděpodobnost nalezení důkazu neekvivalence alespoň $1-\frac{1}{2^k}$. □

Z hlediska počtu potřebných náhodných bitů je použití větší množiny M efektivnější postup zvýšení spolehlivosti než opakování. Například při $|M|=2^k n$ dostaneme pravděpodobnost nalezení důkazu neekvivalence alespoň

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

s využitím $(k+\log_2 n)n$ náhodných bitů.

Problém neekvivalence read-once rozhodovacích diagramů je self-reducibilní. K důkazu zkonstruujme transformaci požadovanou v definici self-reducibility následovně. Pokud diagramy nemají společnou proměnnou (například jeden nebo oba jsou konstantní), pak jsou diagramy ekvivalentní právě tehdy, když jsou oba rovny téže konstantě. To lze zjistit v polynomiáním čase a definujeme hodotu transformace jako 0 nebo 1 podle správné odpovědi. Pokud mají diagramy společnou proměnnou, převedeme test jejich ekvivalence na dva testy ekvivalence po dosazení hodnoty 0 a 1 za libovolnou ze společných proměnných. Kódování diagramů musí být takové, aby fixace libovolné proměnné snížila délku zápisu.

Protože test neekvivalence read-once rozhodovacích diagramů je self-reducibilní a je řešitelný polynomiálním pravděpodobnostním algoritmem ve třídě BPP, má také polynomiální obvody. Pak z Věty 5.2 dostaneme, že pravděpodobně není NP-úplný, protože jinak by se polynomiální hierarchie rovnala své druhé hladině.

8 Pravděpodobnostní Turingův stroj

Definice 8.1 *Pravděpodobnostní TS (PTS) je TS s přidanou páskou, na níž je před zahájením výpočtu zapsána posloupnost náhodných bitů, jejíž délka je dostatečná k tomu, aby čtecí hlava na této pásce během výpočtu nikdy nedošla za poslední předem připravený náhodný bit. Počet náhodných bitů budeme vyjadřovat jako funkci $s(n)$, kde n je délka vstupního slova, a budeme předpokládat, že při libovolném vstupu w bude přečteno nejvýše $s(|w|)$ náhodných bitů. PTS, který pracuje v čase polynomiálním od délky vstupu, budeme nazývat polynomiální PTS a označovat jako PPTS.*

PTS může při svém výpočtu využívat náhodné bity, proto může mít pro jedno vstupní slovo w více výpočtů. 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 porovnání si uveďme, že kdyby přijetí záviselo pouze na tom, zda existuje kombinace náhodných bitů, která vede k přijetí w, dostali bychom TS ekvivalentní nedeterministickému TS.

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

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

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 $0 \le \varepsilon(n) < \frac{1}{2}$ je libovolná reálná funkce.

Definice 8.3 Definujme třídy RP, co-RP a BPP tak, že pro každý jazyk L platí:
(i) $L \in$ RP $\iff$ existuje $(0, \frac{1}{2})$ – PPTS pro L;
(ii) $L \in$ co-RP $\iff \setminus L \in$ RP;
(iii) $L \in$ BPP $\iff$ existuje $(\frac{1}{3}, \frac{2}{3})$ – PPTS 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.

Protože pro jazyky z RP je $\alpha(n) = 0$, platí, že pro dané vstupní slovo existuje alespoň jeden přijímající výpočet právě tehdy, když slovo do přijímaného jazyka patří. 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 na rozdíl od třídy 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 PPTS přijímající jazyk z RP vydá chybný výsledek, lze snížit na exponenciálně malou hodnotu.

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

Nyní dokážeme tvrzení o zesilování pravděpodobnosti správné odpovědi. Nejjednodušší zesílení se dostane 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 8.4 *Nechť $q(n)$ je libovolný polynom. Pak pro každý jazyk L platí:*
(i) Jestliže existuje $\left(0, \frac{1}{q(n)}\right)$-PPTS pro L, pak $L \in$ RP;
(ii) Jestliže $L \in$ RP, pak existuje $\left(0, 1-\left(\frac{1}{2}\right)^{q(n)}\right)$-PPTS 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 $(0, 1 - \mathrm{e}^{-\varepsilon(n)k(n)})$-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 $1 - x \leq \mathrm{e}^{-x}$, která platí pro každé reálné x, dostaneme, že pravděpodobnost zamítnutí je pro M' nejvýše

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

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

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

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

Věta 8.5 *Nechť $q(n)$ je libovolný polynom. Pak pro každý jazyk L platí:*
(i) Jestliže $\beta(n) - \alpha(n) \geq \frac{1}{q(n)}$ a jestliže existuje $(\alpha(n), \beta(n))$-PPTS pro L, pak $L \in$ BPP;
(ii) Jestliže $L \in$ BPP, pak existuje $\left(\left(\frac{1}{2}\right)^{q(n)}, 1 - \left(\frac{1}{2}\right)^{q(n)}\right)$-PPTS pro L.

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 $p \in [0, 1]$. Připomeňme, že $\mathrm{E}X$ označuje střední hodnotu X. Protože $X \in \{0, 1\}$, platí $\mathrm{E}X = \Pr(X = 1) = p$. Nechť X_i pro $i = 1, 2, \ldots, N$ jsou nezávislé realizace veličiny X. Pak pro každé $\Delta \geq 0$ platí

$$\Pr\left(\frac{1}{N}\sum_{i=1}^{N} X_i \geq \mathrm{E}X + \Delta\right) \leq \mathrm{e}^{-2\Delta^2 N}$$

a

$$\Pr\left(\frac{1}{N}\sum_{i=1}^{N} X_i \leq \mathrm{E}X - \Delta\right) \leq \mathrm{e}^{-2\Delta^2 N}.$$

Nechť M je $(\alpha(n), \beta(n))$-PPTS pro nějaký jazyk L. Pro důkaz (i) i (ii) zkonstruujeme M', který pro libovolný vstup w několikrát nezávisle zopakuje M. Počet

opakování $N(w)$ zvolíme pro každé z tvrzení (i) a (ii) zvlášť. 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 $\frac{1}{N(w)}\sum_{i=1}^{N(w)} X_i \geq \frac{\alpha(|w|)+\beta(|w|)}{2}$.

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

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

Vypočtěme ještě pravděpodobnost zamítnutí pro $w \in L$. V tomto případě je $\mathrm{E}X \geq \beta$. Je tedy

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

V případě $w \in L$ je tedy pravděpodobnost přijetí alespoň $1 - \mathrm{e}^{-\frac{1}{2}(\beta-\alpha)^2 N}$. Stroj M' je tedy $(\mathrm{e}^{-\frac{1}{2}(\beta-\alpha)^2 N}, 1 - \mathrm{e}^{-\frac{1}{2}(\beta-\alpha)^2 N})$-PPTS.

Pro důkaz (i) použijeme počet opakování $N(w) = 4q^2(|w|)$. Protože v tomto případě předpokládáme, že $\beta(n) - \alpha(n) \geq \frac{1}{q(n)}$, dostaneme

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

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

Pro důkaz (ii) necháme M' opakovat M nezávisle $N(w) = 16q(|w|)$ krát. V tomto případě je $\alpha = \frac{1}{3}$ a $\beta = \frac{2}{3}$, a tedy

$$\mathrm{e}^{-\frac{1}{2}(\beta(|w|)-\alpha(|w|))^2 N(|w|)} \leq \mathrm{e}^{-\frac{1}{18}N(w)} = \mathrm{e}^{-\frac{8}{9}q(|w|)} \leq \left(\frac{1}{2}\right)^{q(|w|)},$$

protože $\mathrm{e}^{8/9} \geq 2$. Stroj M' je tedy $(\left(\frac{1}{2}\right)^{q(|w|)}, 1 - \left(\frac{1}{2}\right)^{q(|w|)})$-PPTS, jak bylo zapotřebí pro (ii). □

K odvození důsledku Věty 8.5 pro vztah BPP a P/poly zaveďme následující pojem.

Definice 8.6 Nechť L je jazyk z BPP a nechť $R(w,r)$ je relace popisující přijetí w pomocí některého PPTS pro L. Pak slovo r nazveme dobrá nápověda pro 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 8.7 *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 n existuje slovo r_n délky polynomiální v n, které je dobrou nápovědou pro slova délky n v jazyce L.*

Důkaz. Zvolme $q(n) = n^2$. Podle bodu (ii) z Věty 8.5 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 PPTS a nechť $s(n)$ je délka použitých posloupností náhodných bitů r využitých při jeho výpočtu. Pro každé slovo w označme S_w množinu $r \in \{0,1\}^{s(n)}$, která splňují $w \in L \Leftrightarrow \neg R(w,r)$. Slova $r \in S_w$ jsou tedy ty kombinace náhodných bitů, které vedou ke špatné odpovědi pro w. Protože vycházíme z $(\varepsilon(n), 1-\varepsilon(n))$-PPTS pro L, dostaneme, že pro každé w délky n platí

$$|S_w| \leq \varepsilon(n) 2^{s(n)}.$$

Všimněme si, že k důkazu věty stačí ukázat, že pro každé dost veliké n existuje slovo $r_n \in \{0,1\}^{s(n)}$, které nepatří do S_w pro žádné slovo w délky n. Takové slovo je pak dobrou nápovědou pro danou délku n. Jestliže abeceda jazyka L je Σ, pak pro každé dost velké n platí

$$\left| \bigcup_{|w|=n} S_w \right| \leq |\Sigma|^n \varepsilon(n) 2^{s(n)} \leq \frac{|\Sigma|^n}{2^{n^2}} 2^{s(n)} < 2^{s(n)} \ .$$

Z toho plyne, že požadované slovo r_n existuje pro každé dost velké n a věta je tedy dokázána. □

Z dokázané věty plyne, že libovolný jazyk z BPP, tedy také z RP nebo z co-RP, je rozpoznatelný (neuniformními) polynomiálními obvody.

Věta 8.8 *Platí inkluze* BPP $\subseteq$ P/poly.

Důkaz. Nechť $L \in$BPP. 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 $z \in R$ při nějakém 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 8.7 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. □