Reprezentace Booleovských funkcí

Booleovská funkce n proměnných je funkce $f : \{0,1\}^n \to \{0,1\}$. Reprezentace Booleovské funkce je libovolné kombinatorická struktura, která funkci vyjadřuje. Příkladem reprezentace je tabulka funkce, tj. výčet hodnot $f(x)$ pro všechny prvky $x \in \{0,1\}^n$. Tato reprezentace obsahuje vždy 2^n hodnot, a proto je pro větší n velmi neúsporná. Jiný příklad reprezentace je Booleovská formule nebo obvod. Pro funkce, ve kterých lze nalézt určité pravidelnosti, může být tato reprezentace podstatně menší než 2^n. Na druhé straně, protože Booleovských funkcí n proměnných je 2^{2^n}, tak při jakémkoli způsobu reprezentace existují funkce, jejichž reprezentace vyžaduje alespoň 2^n bitů ke svému zápisu.

Pro každý model pro reprezentaci Booleovských funkcí zavedeme míru složitosti (velikosti) reprezentace. Složitostí funkce pak bude minimální složitost její reprezentace v daném modelu. Při porovnávání různých výpočetních modelů pro reprezentaci Booleovských funkcí budeme jejich vyjadřovací sílu porovnávat tím, že budeme porovnávat množiny funkcí polynomiální složitosti.

Kromě Booleovských formulí a obvodů budeme mluvit ještě o reprezentaci Booleovských funkcí pomocí rozhodovacích diagramů a jejich podtříd. Z těchto podtříd se budeme zabývat read-once rozhodovacími diagramy, uspořádanými rozhodovacími diagramy, které jsou známy jako OBDD (ordered binary decision diagrams), a rozhodovacími stromy.

1 Základní pojmy, disjunktivní a konjunktivní normální forma

Booleovská krychle dimenze n je množina $\{0,1\}^n$, jejíž prvky odpovídají všem možným ohodnocením proměnných $x_1, \ldots, x_n \in \{0,1\}$. Prvky krychle budeme někdy nazývat body krychle. Jsou to vektory délky n, jejichž složky budeme nazývat bity. Nulový, resp. jedničkový, vektor budeme obvykle zapisovat jako 0^n, resp. 1^n. Dále, pro každé $i = 1, \ldots, n$ budeme symbolem e_i rozumět vektor, který má jedničku na i-té souřadnici a nulu na ostatních souřadnicích.

Pro body krychle x, y definujeme Hammingovskou vzdálenost $\rho(x, y)$ jako počet souřadnic, na kterých se vektory x a y liší. Body krychle, které mají Hammingovskou vzdálenost 1 budeme nazývat sousední. Váhou vektoru budeme rozumět

počet jeho jedničkových souřadnic. Váhu vektoru x budeme značit $|x|$. Platí, že $|x| = \sum_{i=1}^{n} x_i = \rho(0, x)$.

Booleovská krychle tvoří svaz, pokud zavedeme částečné uspořádání $x \leq y$ právě když $x_i \leq y_i$ pro $i = 1, \ldots, n$. Příslušné svazové operace jsou konjunkce a disjunkce vektorů po jednotlivých bitech. Vektory složené z 0 a 1 lze reprezentovat také pomocí množiny indexů, kde má vektor hodnotu 1. V tomto případě odpovídají svazové operace průniku a sjednocení těchto množin.

Booleovskou funkci f n proměnných nazveme monotonní, pokud pro libovolné dva vstupy x, y délky n, pro které platí $x \leq y$, je $f(x) \leq f(y)$.

Počet $M(n)$ monotonních Booleovských funkcí n proměnných zkoumal Dedekind již v roce 1897 a dnes jsou známy i velmi přesné odhady. Následující odhad není nejpřesnější známý odhad, ale přesnější odhady jsou podstatně složitější.

Věta 1.1 (Kleitman, 1969) *Pro $n \to \infty$ platí*

$$M(n) = 2^{(1+o(1))\binom{n}{\lfloor n/2 \rfloor}} = 2^{\Theta(2^n/\sqrt{n})} .$$

Pro speciální účely se krychle někdy chápe jako lineární prostor dimenze n nad dvouprvkovým tělesem. Sčítání v tomto tělese je sčítání modulo 2, které se značí $\oplus$ a někdy se též nazývá XOR (exclusive or). Násobení odpovídá konjunkci.

Lineární booleovské funkce jsou funkce tvaru $f(x) = a_1 x_1 \oplus \ldots \oplus a_n x_n \oplus b$, kde $a_i, b \in \{0, 1\}$, tedy jde o lineární funkce ve dvouprvkovém tělese. Speciálně, součet všech proměnných, tedy funkci $f(x) = x_1 \oplus \ldots \oplus x_n$, budeme nazývat parita, protože její hodnota závisí na tom, zda je počet jedniček v jejím argumentu sudý nebo lichý.

Podkrychlí rozumíme podmnožinu krychle, která je definována tak, že některé souřadnice fixujeme na konstanty a ostatní ponecháme bez omezení. Takovéto dosazení za některé proměnné budeme nazývat částečný vstup. Částečné vstupy je výhodné zapisovat jako posloupnosti symbolů z $\{0, 1, *\}^n$, ve kterých znak $*$ znamená souřadnice, které nejsou fixovány. Z uvedené reprezentace částečných vstupů plyne, že počet podkrychlí v krychli dimenze n je právě 3^n.

Jestliže $f(x_1, \ldots, x_n)$ je Booleovská funkce, pak funkcí duální nazýváme funkci $f^*(x_1, \ldots, x_n) = \neg f(\neg x_1, \ldots, \neg x_n)$. Například $(x \vee yz)^* = x(y \vee z)$. Pro každou funkci f platí $(f^*)^* = f$.

Pro každou Booleovskou funkci f budeme označením $\mathrm{Supp}(f)$ rozumět množinu těch vstupů x, pro které je $f(x) = 1$.

Řekneme, že funkce podstatně závisí na některé proměnné, jestliže dosazením 0 a 1 za tuto proměnnou dostaneme různé funkce zbylých proměnných.

1.1 Disjuntivní normální forma

Monomem (termem) budeme nazývat konjunkci několika konsistentních literálů, přičemž literál je proměnná nebo její negace. Počet literálů v monomu nazýváme

jeho délkou. Disjunkci množiny monomů budeme nazývat disjunktivní normální formou (DNF) pro zápis Booleovské funkce. Složitostí DNF budeme rozumět počet monomů, ze kterých se skládá. Pro libovolnou funkci f budeme jako $\mathrm{dnf}(f)$ označovat minimální složitost DNF, která funkci f vyjadřuje.

Monom délky 0, tedy konjunkci prázdné množiny literálů, považujeme za konstantu 1. Disjunkci prázdné množiny monomů považujeme za konstantu 0. Obě konstantní funkce tedy mají vyjádření pomocí DNF.

Pro zjednodušení zápisu monomů budeme negaci reprezentovat pomocí vodorovné čáry nad příslušným literálem a konjunkci budeme vynechávat. Tedy například, místo $\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4$ budeme psát $\overline{x_1}x_2\overline{x_3}x_4$.

Splňující ohodnocení proměnných pro daný monom je takové ohodnocení, při kterém má monom hodnotu 1. Množina splňujících ohodnocení monomu je podkrychle a lze ji tedy reprezentovat pomocí částečného vstupu tak, že proměnné, které se v monomu vyskytují, mají přiřazenu hodnotu 0 nebo 1, která splňuje literál obsahující danou proměnnou. Proměnné, které se v monomu nevyskytují, jsou reprezentovány $*$.

Jako podmonom monomu m označíme každý monom, který vznikne z m odstraněním některých literálů. Podmonom nazveme vlastní, pokud vznikl odstraněním alespoň jednoho literálu.

Monom nazveme úplný, pokud obsahuje literál od každé proměnné. Takový monom je splněn jen jedním ohodnocením všech proměnných.

Monom m nazveme implikant funkce f, jestliže pro každý vstup x platí $m(x) \Rightarrow f(x)$ nebo, ekvivalentně, $m(x) \leq f(x)$. Monom je implikant f právě tehdy, když může být členem (termem) DNF pro funkci f.

Monom m nazveme primární implikant (někdy též minterm) funkce f, jestliže je implikantem f, ale žádný vlastní podmonom m' monomu m již není implikantem f. Je-li m implikant funkce f, pak existuje primární implikant m' funkce f, který vznikne z m odstraněním některých (případně všech) literálů.

Každou Booleovskou funkci lze vyjádřit pomocí DNF, protože můžeme ke každému bodu ze $Supp(f)$ vzít úplný monom, který je splněn právě v tomto jednom bodě. Disjunkce těchto monomů přes všechny body $Supp(f)$ dává DNF pro f.

Věta 1.2 *Pro libovolnou Booleovskou funkci existuje minimální DNF složená pouze z primárních implikantů.*

Důkaz. Zvolme libovolnou minimální DNF pro f a nahraďme každý její monom některým jeho podmonomem, který je primárním implikantem f. Touto náhradou nemůže počet monomů v DNF vzrůst a vyjadřovaná funkce se nezmění. □

Důsledek 1.3 *Každá Booleovská funkce je disjunkcí svých primárních implikantů.*

Důkaz. Vezměme minimální DNF složenou jen z primárních implikantů. Pokud neobsahuje všechny primární implikanty, můžeme je do DNF přidat, protože tím se vyjádřená funkce nezmění. □

Pro obecnou funkci nemusí minimální DNF obsahovat všechny primární implikanty. Například disjunktivní normální formy $\overline{x_1}x_2 \vee \overline{x_2}x_3 \vee \overline{x_3}x_1$ a $x_1\overline{x_2} \vee x_2\overline{x_3} \vee x_3\overline{x_1}$ reprezentují tutéž funkci, která má 6 primárních implikantů tvořených sjednocením množin monomů použitých v obou reprezentacích.

Následující věta ukazuje maximální složitost disjunktivní normální formy.

Věta 1.4 *Každá DNF, která vyjadřuje funkci parita n proměnných, má složitost alespoň 2^{n-1}. Na druhé straně, každou funkci n proměnných lze vyjádřit pomocí DNF složitosti nejvýše 2^{n-1}.*

Důkaz. DNF pro paritu může obsahovat pouze úplné monomy, protože pokud monom neobsahuje všechny proměnné, pak je splněn na některých dvou sousedních bodech krychle, které mají různou paritu. DNF, která je složena jen z úplných monomů musí obsahovat právě tolik monomů, kolik má daná funkce splňujících ohodnocení, tedy na kolika bodech krychle má hodnotu 1. Protože parita má právě 2^{n-1} bodů s hodnotou 1, je toto také složitost její DNF.

Ukážeme ještě, že každou funkci f n proměnných lze vyjádřit pomocí DNF složitosti nejvýše 2^{n-1}. Sdružíme body krychle do dvojic, které se liší jen v hodnotě jedné proměnné, například x_1. Pokud nejvýše jeden z bodů některé dvojice má hodnotu 1, pak jej reprezentujeme úplným monomem. Pokud oba body ve dvojici mají hodnotu 1, reprezentujeme funkci na obou současně monomem, ve kterém se nevyskytuje x_1 a který je na bodech zvolené dvojice roven 1 a na všech ostatních bodech krychle je roven 0. Tímto způsobem funkci reprezentujeme tak, že ke každé dvojici bodů potřebujeme jen jeden monom a je jich tedy potřeba nejvýše 2^{n-1}. □

Dále se budeme zabývat DNF pro monotonní funkce.

Věta 1.5 *Pro libovolnou monotonní funkci f platí*

$$f(x) = \bigvee_{f(a)=1} \bigwedge_{a_i=1} x_i \tag{1}$$

a tedy každou monotonní funkci lze vyjádřit jako disjunkci monotonních monomů.

Důkaz. Pro libovolné vektory $a, x \in \{0,1\}^n$ platí

$$a \leq x \Leftrightarrow \bigwedge_{a_i=1} x_i = 1 \ .$$

Pokud je tedy $f(x) = 1$, pak člen DNF v pravé straně (1) odpovídající $a = x$ je také roven 1 a identita platí. Pokud je některý člen DNF roven 1 pro x, pak

vektor a, který tento člen určuje, splňuje $f(a) = 1$ a $a \leq x$. Z monotonnosti f plyne $f(x) = 1$ a identita (1) je tedy splněna i v tomto případě. □

Lemma 1.6 *Každý primární implikát monotonní funkce je monotonní.*

Důkaz. Důkaz provedeme sporem. Nechť m je nemonotonní primární implikát monotonní funkce f. Bez újmy na obecnosti nechť je tvaru $m = (\neg x_1) \wedge m'$. Protože

$$(\neg x_1) \wedge m' \Rightarrow f(x_1, \ldots, x_n)$$

platí také

$$m' \Rightarrow f(0, x_2, \ldots, x_n).$$

Z monotonie f vyplývá

$$f(0, x_2, \ldots, x_n) \Rightarrow f(1, x_2, \ldots, x_n)$$

Celkem tedy dostaneme, že

$$m' \Rightarrow f(x_1, \ldots, x_n).$$

To znamená, že m' je term, což je spor s předpokladem, že m je primární implikát. □

Věta 1.7 *Pro monotonní funkci f je* $\mathrm{dnf}(f)$ *právě rovné počtu primárních implikantů. Navíc, existuje právě jedna reprezentace, která je složena pouze z primárních implikantů, a to je disjunkce všech primárních implikantů.*

Důkaz. Disjunkce všech primárních implikantů f je rovna funkci f. Z toho plyne, že $\mathrm{dnf}(f)$ je nejvýše počet primárních implikantů f.

K důkazu opačné nerovnosti zvolme minimální DNF ϕ pro f tvořenou jen primárními implikanty f. Dokážeme, že ϕ nutně obsahuje všechny primární implikanty f.

Každému primárnímu implikantu m funkce f přiřaďme vstup $a_m \in \{0,1\}^n$, který přiřazuje jedničku právě těm proměnným, které se v m vyskytují. Protože podle Lemmatu 1.6 je m monotonní, je $m(a_m) = 1$. Z monotonie primárních implikantů plyne také to, že žádné dva primární implikanty f nemají porovnatelné množiny proměnných, které se v nich vyskytují. Proto je m jediným primárním implikantem f, který je splněn vstupem a_m.

Kdyby ϕ neobsahovala některý primární implikant m, neobsahovala by žádný monom, který by byl vstupem a_m splněn a bylo by tedy $\phi(a_m) = 0$. To by byl spor s tím, že $f(a_m) = 1$. Dokázali jsme, že ϕ obsahuje všechny primární implikanty f. Protože délka ϕ je rovna $\mathrm{dnf}(f)$, je $\mathrm{dnf}(f)$ alespoň počet primárních implikantů f. Tím je věta dokázána. □

Poznamenejme, že pokud nebudeme požadovat, aby členy DNF pro monotonní funkci byly primární implikanty, pak mohou existovat i nemonotonní minimální DNF pro danou funkci. Například pro funkci disjunkce, jejíž minimální DNF z primárních implikantů je $x_1 \vee x_2$, patří mezi minimální DNF také $x_1 \vee \overline{x_1} x_2$.

1.2 Konjunktivní normální forma

Duálním pojmem k DNF je konjunktivní normální forma (conjunctive normal form, CNF). Formule ve tvaru CNF je konjunkcí klausulí, z nichž každá je disjunkcí konzistentních a různých literálů. Složitostí CNF budeme rozumět počet jejích klausulí. Zápisem $\mathrm{cnf}(f)$ budeme rozumět minimální složitost CNF vyjadřující funkci f.

DNF a CNF jsou navzájem duální v tom smyslu, že záměnou konjunkcí a disjunkcí v DNF pro libovolnou funkci f dostaneme CNF pro funkci f^*. Jednodušší převod mezi DNF a CNF plyne z toho, že formální negací DNF pro f dostaneme CNF pro $\neg f$.

Pokud vyjdeme z minimální DNF pro f, dostaneme její negací CNF pro $\neg f$. Platí tedy $\mathrm{cnf}(\neg f) \leq \mathrm{dnf}(f)$. Podobnou úvahu můžeme použít pro CNF místo DNF a pro duální funkci místo negace. Celkem dostaneme, že pro libovolnou f platí

$$\begin{aligned} \mathrm{dnf}(\neg f) &\leq \mathrm{cnf}(f) \\ \mathrm{dnf}(f^*) &\leq \mathrm{cnf}(f) \\ \mathrm{cnf}(\neg f) &\leq \mathrm{dnf}(f) \\ \mathrm{cnf}(f^*) &\leq \mathrm{dnf}(f) \end{aligned}$$

Použitím těchto nerovností pro f a pro $\neg f$ dostaneme

$$\mathrm{dnf}(f) = \mathrm{cnf}(\neg f) = \mathrm{cnf}(f^*)$$

a

$$\mathrm{cnf}(f) = \mathrm{dnf}(\neg f) = \mathrm{dnf}(f^*)\ .$$

Výsledky týkající se DNF lze těmito transformacemi přenést na CNF.

Duálním pojmem k pojmům implikant a primární implikant, které jsou důležité pro DNF, jsou pojmy implikát a primární implikát. Implikát funkce f je klausule c, která splňuje $f \Rightarrow c$ nebo, ekvivalentně, $f \leq c$. Primární implikát je implikát, který je minimální v podobném smyslu jako primární implikant, totiž že vypuštěním libovolného literálu dostaneme klausuli, která není implikátem.

Monom m je (primárním) implikantem f právě tehdy, když klausule $c = \neg m$ je (primárním) implikátem $\neg f$, a také právě tehdy, když klausule $c' = m^*$ je (primárním) implikátem f^*

2 Modely pro reprezentaci Booleovských funkcí

Existují dva hlavní typy modelů pro reprezentaci Booleovských funkcí. První typ jsou modely založené na kombinování Booleovských hodnot pomocí Booleovských spojek. K tomuto typu patří různé varianty obvodů a formulí. Druhý typ jsou různé varianty rozhodovacích diagramů.

2.1 Obvody a formule

Formule a obvody jsou založené na kombinování funkcí pomocí Booleovských spojek. Při definici konkrétní třídy formulí nebo obvodů je třeba uvést, které Booleovské funkce jsou povoleny jako spojky. Tuto množinu spojek budeme nazývat bází. Může to být libovolná konečná množina Booleovských funkcí. Nejčastěji budeme používat báze $\{0, 1, \neg, \vee, \wedge\}$ a $\{0, 1, \oplus, \wedge\}$.

Formulí nebo obvodem lze vyjádřit jen takové funkce, které vzniknou skládáním funkcí ze zvolené báze. Byly studovány podmínky, za kterých je báze spojek úplná, tj. dovoluje vyjádřit každou funkci. Tyto podmínky zformulujeme v sekci 6. Formule a obvody při stejné bázi se mohou lišit velikostí, ale množiny reprezentovatelných funkcí jsou stejné.

Speciálním případem formule jsou disjunktivní normální forma (DNF) a konjunktivní normální forma (conjunctive normal form, CNF). Pro měření velikosti reprezentace v těchto dvou modelech budeme používat počet monomů resp. klausulí.

Složitost obvodu je počet jeho uzlů (hradel) realizujících Booleovské spojky. Booleovská formule je výraz složený z proměnných a spojek. Za velikost formule považujeme počet všech výskytů proměnných. Minimální velikost obvodu pro f v bázi B budeme značit $C_B(f)$ a minimální velikost formule $L_B(f)$. Pokud je báze zřejmá z kontextu, nebudeme ji ve značení uvádět.

Věta 2.1 *Každou Booleovskou funkci n proměnných lze vyjádřit formulí v bázi $\{0, 1, \neg, \vee, \wedge\}$ velikosti $O(2^n)$.*

Důkaz. Označme jako s_n maximum velikosti formule potřebné pro vyjádření libovolné funkce f n proměnných. Protože konstanty jsou prvky báze, je $s_0 = 0$. Protože libovolnou funkci n proměnných lze vyjádřit jako $f(x_1, \ldots, x_n) = \neg x_1 \wedge f(0, x_2, \ldots, x_n) \vee x_1 \wedge f(1, x_2, \ldots, x_n)$, platí $s_n \leq 2\, s_{n-1} + 2$. Z toho indukcí plyne $s_n \leq 2^{n+1} - 2$. □

Z věty plyne také vyjádření funkce obvodem velikosti $O(2^n)$. Později, konkrétně pomocí Vět 4.1 a 3.1 ukážeme, že každou funkci n proměnných lze reprezentovat obvodem velikosti $O(2^n/n)$.

Pro konstrukci obvodů se někdy hodí jejich vyjádření pomocí lineárního programu. Tím se obecně myslí posloupnost přiřazovacích příkazů, které se postupně provedou v pořadí, jak jsou zapsány. Pro reprezentaci obvodů lze použít lineární programy, které obsahují pouze Booleovské výrazy, vychází z množiny vstupních proměnných a každý příkaz přiřadí hodnotu proměnné, která dosud ohodnocena nebyla, pomocí výrazu s proměnnými, které již ohodnoceny byly. Pokud všechny výrazy v programu obsahují jen jednu spojku, pak je počet příkazů roven velikosti obvodu, tj. počtu použitých spojek. Konstrukce obvodu z lineárního programu v tomto případě přiřadí každému příkazu jeden uzel obvodu ohodnocený spojkou použitou v tomto příkazu. Hrany do vytvořeného uzlu povedou z uzlů, které vypočítávají proměnné, které jsou spojkou kombinovány.

Opačnou konstrukci, kdy z obvodu chceme vytvořit lineární program, lze provést tak, že uzly obvodu očíslujeme tak, aby předchůdci každého uzlu měly číslo menší než daný uzel. Protože obvod je acyklický graf, lze takové pořadí nalézt. Každému uzlu obvodu přiřadíme novou proměnnou. Lineární program bude hodnoty těchto proměnných přiřazovat ve zvoleném pořadí uzlů. Tím dosáhneme toho, že všechny proměnné potřebné v některém příkazu mají hodnotu přiřazenu některým z předchozích příkazů.

2.2 Rozhodovací diagramy

V tomto paragrafu popíšeme deterministické rozhodovací diagramy, které v každém větvícím uzlu testují jednu proměnnou. Kromě obecných rozhodovacích diagramů budeme zkoumat jejich podtřídy, konkrétně read-once rozhodovací diagramy, uspořádané rozhodovací diagramy (OBDD) a rozhodovací stromy.

Rozhodovací diagram (branching program) pro Booleovskou funkci $f(x_1, \ldots, x_n)$ je acyklický orientovaný graf s jedním počátečním uzlem, jehož koncové uzly jsou ohodnoceny 0,1. Každý nekoncový uzel je ohodnocen některou vstupní proměnnou a má dva následníky nazývané 0-následník a 1-následník, které jsou rozlišené ohodnocením příslušných hran. Výpočet pro daný vstup x začíná v počátečním uzlu a v každém nekoncovém uzlu je testována proměnná, která je uzlu přiřazena. Výpočet pak pokračuje do 0-následníka nebo 1-následníka podle hodnoty testované proměnné. Ohodnocení koncového uzlu, kde skončí výpočet, je $f(x)$.

Rozhodovací diagram se nazývá read-once, jestliže každý výpočet testuje každou proměnnou nejvýše jednou. Jestliže existuje uspořádání proměnných π takové, že všechny výpočty v daném diagramu testují proměnné v pořadí, které je konzistentní s π (nemusí testovat všechny proměnné), pak se diagram nazývá uspořádaný. Pro tyto diagramy se používá označení π-OBDD (ordered binary decision diagram), nebo jen OBDD, pokud není konkrétní uspořádání podstatné. Rozhodovací diagram, jehož graf je strom, se nazývá rozhodovací strom.

Velikost rozhodovacího stromu definujeme jako počet jeho listů. Pro ostatní varianty rozhodovacích diagramů definujeme velikost jako počet všech uzlů diagramu.

Složitost funkce f při vyjadřování pomocí rozhodovacích stromů budeme značit $\mathrm{dt}(f)$ (decision tree), při vyjadřování pomocí read-once rozhodovacích diagramů 1-$\mathrm{bdd}(f)$ (decision diagram). Složitost f při reprezentaci π-OBDD budeme značit π-$\mathrm{OBDD}(f)$ a minimum přes všechna možná pořadí π jako $\mathrm{OBDD}(f)$.

Věta 2.2 *Každou Booleovskou funkci n proměnných lze vyjádřit rozhodovacím stromem velikosti 2^n.*

Důkaz. Proměnné jsou testovány na všech větvích ve stejném pořadí. Na hladině i, tj. po otestování i proměnných, je 2^i uzlů. Strom má tedy 2^n listů, což jsme měli dokázat. □

Z toho plyne také vyjádření funkce pomocí (read-once) rozhodovacího diagramu velikosti $2^{n+1} - 1$, protože strom je speciálním případem těchto struktur a do velikosti se pro tyto modely započítávají vnitřní uzly.

2.3 Měření složitosti

Pro každý model měříme velikost reprezentace obvykle způsobem, který je pro daný model přirozený. Pro modely, které již byly zavedeny v předchozích sekcích, shrnuje způsob měření velikosti následující tabulka.

Název modelu	míra velikosti	značení
Booleovské obvody	počet uzlů se spojkami	$C()$
rozhodovací diagramy	počet uzlů	bdd()
Booleovské formule	počet výskytů proměnných	$L()$
DNF	počet monomů	dnf()
CNF	počet klausulí	cnf()
read-once rozhodovací diagramy	počet uzlů	1-bdd()
rozhodovací stromy	počet listů	dt()
π-OBDD	počet uzlů	π-OBDD()

Nechť M je model pro reprezentaci Booleovských funkcí a f je Booleovská funkce. Složitostí funkce f v modelu M rozumíme minimální velikost mezi všemi reprezentacemi funkce f v daném modelu.

Pro konkrétní modely budeme pro jednoduchost používat speciální značení odvozené z názvu modelu, které jsme již použili, například dnf(f), dt(f) a pod. Složitost funkce f v obecném modelu M budeme značit $\text{size}_M(f)$.

Tatáž funkce může mít v různých modelech různou velikost. Schopnost modelu vyjádřit funkce úsporným způsobem budeme nazývat vyjadřovací sílou modelu. Cílem našeho zkoumání bude porovnání vyjadřovací síly různých modelů pomocí tříd funkcí, které mají polynomiální složitost. Pro tento účel je potřeba uvažovat nejen reprezentace jednotlivých funkcí, ale také posloupností B.f. rostoucího počtu proměnných. K tomu použijeme posloupnost reprezentací, pro každou funkci posloupnosti jednu. Složitost pak měříme jako závislost velikosti reprezentace na počtu proměnných.

Definice 2.3 Uvažujme posloupnosti Booleovských funkcí $\{f_i\}_{i=1}^{\infty}$, přičemž počet proměnných funkce f_i označme n_i. Pro libovolný model M budeme jako Poly(M) označovat množinu (třídu) všech posloupností funkcí f_i, které pro každé i splňují $n_{i+1} > n_i$ a pro které existuje polynom $p(n)$ tak, že pro každé i je $\text{size}_M(f_i) \leq p(n_i)$.

Při srovnání síly modelů M_1 a M_2 může nastat několik možností, které shrnuje následující tabulka. U názvů jednotlivých vztahů vynecháváme pro stručnost

slovo polynomiálně, tj. např. polynomiálně ekvivalentní, které ale budeme předpokládat vždy, pokud nebude uvedeno jinak.

slovní vyjádření	formální vyjádření
ekvivalentní	$\mathrm{Poly}(M_1) = \mathrm{Poly}(M_2)$
M_2 alespoň tak silný jako M_1	$\mathrm{Poly}(M_1) \subseteq \mathrm{Poly}(M_2)$
M_2 ostře silnější než M_1	$\mathrm{Poly}(M_1) \subsetneq \mathrm{Poly}(M_2)$
M_1 a M_2 jsou neporovnatelné	$\mathrm{Poly}(M_1) \not\subseteq \mathrm{Poly}(M_2)$ a $\mathrm{Poly}(M_2) \not\subseteq \mathrm{Poly}(M_1)$

Vztah M_2 alespoň tak silný jako M_1 budeme někdy také nazývat vnořením M_1 do M_2. Jestliže M_2 je ostře silnější než M_1, budeme také říkat, že model M_2 lze oddělit od M_1.

3 Vzájemné převody uvedených modelů

K důkazu, že model M_2 je alespoň tak silný jako M_1, stačí ukázat, že pro každou funkci n proměnných, která má v M_1 reprezentaci velikosti s, existuje reprezentace v M_2, která má velikost nejvýše polynomiální v s a n. Většina převodů mezi modely v tomto textu je dokonce taková, že velikost nové reprezentace je $O(s)$. Takovéto převody budeme nazývat lineární.

Věta 3.1 *Ke každému rozhodovacímu diagramu velikosti s existuje ekvivalentní obvod velikosti $O(s)$.*

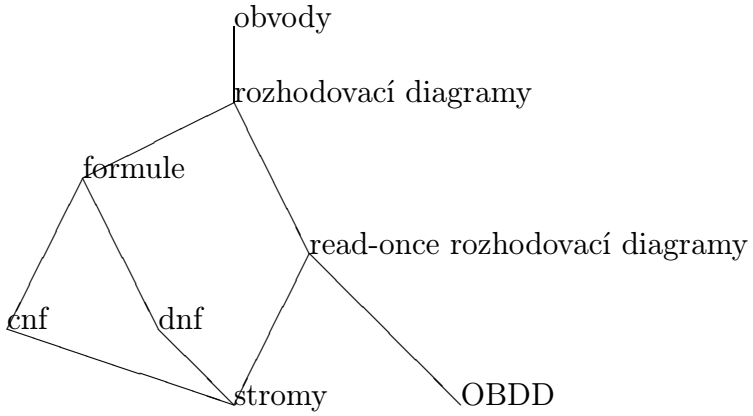
Důkaz. Ke každém uzlu v diagramu definujeme funkci f_v, kterou počítá diagram, jehož počáteční uzel je v. Postupujeme odspodu a ke každému uzlu v diagramu zkonstruujeme obvod, který reprezentuje funkci f_v. Jestliže uzel v testuje proměnnou x_i a má následníky v_0 a v_1 odpovídající v tomto pořadí hodnotám $x_i = 0$ a $x_i = 1$, pak f_v vyjádříme jako $f_v = x_i f_{v_1} \vee \neg x_i f_{v_0}$. Pomocí tohoto vyjádření f_v lze ukázat, že k obvodu, který počítá f_{v_1} a f_{v_0} stačí přidat tři nové uzly se spojkami $\vee$ a $\wedge$ tak, že obvod počítá také f_v. Celková složitost takto vzniklého obvodu je lineární ve velikosti původního rozhodovacího diagramu. □

Věta 3.2 *Ke každé Booleovské formuli v bázi $\{0, 1, \neg, \vee, \wedge\}$ velikosti s existuje ekvivalentní rozhodovací diagram velikosti $O(s)$.*

Důkaz. Postupujeme indukcí podle složitosti ϕ. Jednotlivým proměnným přiřadíme diagramy, jejichž první uzel testuje danou proměnnou a hrany z něj vedou do koncových uzlů s odpovídajícím ohodnocením. Jestliže ϕ je tvaru $\neg\phi_1$, dostaneme diagram pro ϕ z diagramu pro ϕ_1 záměnou hodnot ve výstupních uzlech. Dále, jestiže je ϕ tvaru $\phi_1 \vee \phi_2$ nebo $\phi_1 \wedge \phi_2$, pak nejprve zkonstruujeme diagramy pro funkce ϕ_1 a ϕ_2 a propojíme je tak, aby výsledný diagram počítal disjunkci

nebo konjunkci ϕ_1 a ϕ_2 podle tvaru ϕ. Toto propojení lze udělat tak, že počet uzlů, které testují proměnné ve výsledném diagramu, je součet počtů těchto uzlů v propojovaných diagramech. Ve všech výše uvedených krocích získáme pro formuli ekvivalentní diagram, který obsahuje právě tolik testovacích uzlů, kolik je složitost původní formule. Po započtení výstupních uzlů bude velikost výsledného diagramu pro ϕ rovna $L(\phi) + 2$. □

Obvody, rozhodovací diagramy a formule tedy tvoří posloupnost modelů, ve které každý model zahrnuje modely následující. Další modely studované v tomto textu již nelze seřadit do podobně uspořádané posloupnosti. Vzájemné vztahy mezi některými z těchto modelů vyjadřuje následující diagram. Pokud jsou dva modely propojeny hranou, znamená to, že výše umístěný model je alespoň tak silný jako níže umístěný model.



Kromě převodů dokázaných ve Větách 3.1 a 3.2 je většina dalších převodů v diagramu jen vnoření speciálního případu do obecnějšího modelu, v podstatě při zachování velikosti. Výjimku tvoří převod stromů na DNF a CNF, při kterém dostaneme množinu monomů pro DNF z množiny všech přijímajících cest ve stromu a klausule pro CNF z množiny zamítajících cest.

Pro všechna vnoření modelů uvedená v diagramu se předpokládá, že inkluze příslušných tříd polynomiální složitosti je vlastní, tedy že neplatí opačné vnoření. Dokázáno je to ale jen pro vnoření v dolní části diagramu. Přesněji, ostré inkluze nejsou dokázány mezi obvody, rozhodovacími diagramy a formulemi. Ostatní ostré inkluze v tomto diagramu a neporovnatelnost mezi některými z těchto modelů postupně dokážeme.

Budeme se ještě zabývat transformací obvodů a formulí z jedné úplné báze do druhé.

Věta 3.3 *Pokud jsou B_1 a B_2 dvě úplné báze, pak pro každou f platí*

$$C_{B_2}(f) = O(C_{B_1}(f))$$
$$L_{B_2}(f) = L_{B_1}(f)^{O(1)}$$

Důkaz. Pro transformaci obvodu v bázi B_1 do báze B_2 stačí pro každou spojku báze B_1 najít vyjádření obvodem v bázi B_2 a pak systematicky nahradit všechny spojky báze B_1 těmito obvody v B_2. Protože spojky nahrazujeme obvody konstantní velikosti, zvětší se počet spojek v obvodu nejvýše lineárně.

Pro formule je transformace podobně jednoduchá, pokud je formule vyvážená. Pro tento účel nazveme formuli velikosti L vyváženou, pokud její hloubka (ve smyslu hloubky stromu) je $O(\log L)$. V takovém případě můžeme opět všechny spojky B_1 nahradit ekvivalentními formulemi v B_2. Jestliže maximální složitost vyjádření spojek B_1 v B_2 je k, pak bude mít výsledná formule velikost $k^{O(\log L)} = L^{O(1)}$.

Pro obecnou formuli, která nemusí být vyvážená, je požadované tvrzení dokázáno pro binární spojky ve Větě 3.5. □

Lemma 3.4 *V každém binárním stromu velikosti l existuje podstrom velikosti $[l/3, 2l/3)$.*

Důkaz. Začneme hledat od kořene. Z uzlu stromu, který určuje podstrom velikosti alespoň $2l/3$, přejdeme do následníka, který určuje větší podstrom. Tento podstrom má velikost alespoň $l/3$. Pokud je menší než $2l/3$, byl nalezen podstrom požadovaný v tvrzení. Jinak pokračujeme výše pospaným postupem. Uvedený postup nalezne požadovaný podstrom po konečném počtu kroků. □

Věta 3.5 *Nechť B_1 a B_2 jsou dvě úplné báze a B_1 obsahuje pouze nejvýše binární spojky. Pak k libovolné formuli ϕ v bázi B_1 existuje ekvivalentní formule ϕ' v bázi B_2 hloubky $O(\log L_{B_1}(\phi))$ a velikosti $L_{B_1}(\phi)^{O(1)}$.*

Důkaz. Z lemmatu 3.4 plyne, že ve ϕ existuje podformule velikosti nejvýše $2/3\, L_{B_1}(\phi)$ taková, že zbytek původní formule má rovněž velikost nejvýše $2/3\, L_{B_1}(\phi)$. Označme některou takovou podformuli ψ_1 a jako $\psi_2(y)$ označme formuli, která vznikne z ϕ náhradou ψ_1 za novou proměnnou y. Platí tedy $\phi = \psi_2(\psi_1)$, kde pro jednoduchost vynecháváme ze zápisu všechny proměnné. Pak použijeme identitu

$$\psi_2(\psi_1) = \mathrm{sel}(\psi_1, \psi_2(0), \psi_2(1)) \ , \tag{2}$$

kde $\mathrm{sel}(x, y, z)$ je definováno pro $x, y, z \in \{0, 1\}$ jako

$$\mathrm{sel}(x, y, z) = \begin{cases} y & \text{pokud } x = 0 \\ z & \text{pokud } x = 1 \end{cases} \ .$$

Označme jako h minimální hloubku formule v B_2, která vyjadřuje funkci sel, přičemž do hloubky nebudeme počítat případná použití negace nebo spojek, jejichž jeden argument je konstanta. Tyto spojky ve vyjádření sel zvyšují složitost výsledné formule jen lineárně a není tedy třeba je brát v úvahu. Například pro bázi $\{0, 1, \neg, \vee, \wedge\}$ můžeme použít

$$\mathrm{sel}(x, y, z) = \neg x \wedge y \vee x \wedge z \tag{3}$$

a pro bázi $\{0, 1, \oplus, \wedge\}$

$$\mathrm{sel}(x, y, z) = (x \oplus 1) \wedge y \oplus x \wedge z \ . \tag{4}$$

V obou těchto případech dostaneme $h = 2$.

Použitím zvoleného vyjádření sel v (2) vyjádříme původní formuli vyjádřili pomocí několika formulí menší velikosti spojené novými spojkami ve stromu, jehož větvení tvoří strom hloubky h. Tento postup rekurzivně opakujeme na tyto menší formule. Tím se tyto formule dále zmenšují, ale roste jejich počet a také se zvětšuje hloubka stromu, který tvoří část výsledné formule složené z nových spojek. Po k opakováních tvoří nové spojky strom hloubky kh, do jehož listů jsou dosazeny části původní formule, které již mají velikost nejvýše $(2/3)^k L_{B_1}(\phi)$. Postup skončí nejvýše po $k = O(\log L_{B_1}(\phi))$ krocích, protože po tomto počtu kroků mají zbývající části původní formule velikost nejvýše 1. Výsledná formule má hloubku $O(\log L_{B_1}(\phi))$ $h = O(\log L_{B_1}(\phi))$, a tedy velikost $a^{kh} = L_{B_1}(\phi)^{O(1)}$, kde a je maximální počet argumentů spojek v bázi B_2. Odhad velikosti samotné, pokud není třeba odhadnout hloubku výsledné formule, lze získat také jako l^k, kde l je složitost sel v bázi B_2. □

4 Odhady maximální složitosti funkcí

Věta 4.1 *Pro libovolnou funkci f n proměnných a libovolné jejich uspořádání π existuje vyjádření f pomocí π-*OBDD *velikosti $O(2^n/n)$.*

Důkaz. Uvažme rozhodovací strom pro f, který ve všech větvích testuje všechny proměnné v pořadí $x_{\pi(1)}, \ldots, x_{\pi(n)}$. Na hladinách $i = 0, \ldots, n-1$ obsahuje strom 2^i uzlů testujících $x_{\pi(i+1)}$ a na hladině n je 2^n listů, z nichž každý definuje hodnotu funkce pro jeden vstup. Tento strom zmenšíme jako π-OBDD tím, že postupně odspodu hledáme na jednotlivých hladinách uzly, které jsou na stejné hladině a počítají stejnou funkci. Pokud takové uzly najdeme, zachováme z nich pouze jeden a přesměrujeme do něj hrany, které vedly do ostatních uzlů počítajících tutéž funkci. Tyto ostatní uzly pak zrušíme.

Protože uzly na hladině i, kde $0 \leq i \leq n$, počítají funkce $n - i$ proměnných, je po uvedené redukci na hladině i nejvýše $\min(2^i, 2^{2^{n-i}})$ uzlů. Velikost získaného

π-OBDD můžeme shora odhadnout součtem odhadů jednotlivých hladin. Platí tedy

$$\pi\text{-OBDD}(f) \leq \sum_{i=0}^{n} \min(2^i, 2^{2^{n-i}}) \ .$$

Pro každé $j < n$ dále platí

$$\sum_{i=0}^{n} \min(2^i, 2^{2^{n-i}}) \leq \sum_{i=0}^{j-1} 2^i + \sum_{i=j}^{n} 2^{2^{n-i}} \leq 2^j + \sum_{k=0}^{n-j} 2^{2^k} \ .$$

Ukážeme, že členy sumy v posledním výrazu lze shora odhadnout geometrickou řadou. Pro každé $x \geq 4$ platí $x/2 \geq \sqrt{x}$. Substitucí $x = 2^{2^k}$ dostaneme pro každé $k \geq 1$ nerovnost

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

Platí tedy

$$\sum_{k=1}^{n-j} 2^{2^k} \leq 2 \cdot 2^{2^{n-j}}$$

a tedy celkem

$$\sum_{i=0}^{n} \min(2^i, 2^{2^{n-i}}) \leq 2^j + 2 \cdot 2^{2^{n-j}} + 2 \ .$$

Volbou $j = n - \lfloor \log n \rfloor + 1$ dostaneme

$$\pi\text{-OBDD}(f) \leq \frac{2^{n+2}}{n} + O(2^{n/2}) = O\left(\frac{2^n}{n}\right) \ .$$

Tím je požadovaný odhad dokázán. □

Jako důsledek dostáváme, že pro každou funkci platí také $C(f) \leq O(2^n/n)$.

Věta 4.2 *Pro každé dostatečně veliké n existuje funkce n proměnných, pro kterou platí $C(f) \geq 2^n/(2n)$.*

Důkaz. Odhadněme nejprve shora počet funkcí n proměnných, které lze vyjádřit pomocí obvodů velikosti s. Uzly se spojkami očíslujeme čísly $n+1, \ldots, n+s$, přičemž čísla $1, \ldots, n$ jsou rezervována jako kódy vstupních proměnných. Každému uzlu se spojkou v obvodu přiřadíme kód, což bude číslo nebo trojice čísel. Uzlům počítajícím negaci přiřadíme index uzlu, ze kterého přichází do daného uzlu vstup. Uzlům počítajícím konjunkci nebo disjunkci přiřadíme trojici. Trojice obsahuje index použité spojky (konjunkce nebo disjunkce), a indexy obou předchůdců daného uzlu. Předchůdcem může být buď proměnná nebo uzel se spojkou. Počet různých kódů uzlů je tedy nejvýše $(n+s) + 2(n+s)^2$. Libovolný obvod velikosti s lze až na izomorfismus vyjádřit posloupností s popsaných kódů.

Ne každá posloupnost těcho kódů reprezentuje správně utvořený obvod, ale pro každou takovouto posloupnost lze snadno zjistit, zda reprezentuje obvod nebo ne.

Počet posloupností popsaného typu je horním odhadem počtu obvodů velikosti nejvýše s a tedy také počtu funkcí, které jsou jimi vyjádřeny. Počet uvažovaných posloupností je nejvýše $((n+s)+2(n+s)^2)^s$. Nechť

$$s = \frac{2^n}{2n} \ .$$

Lze snadno ověřit, že pak platí

$$(n+s)+2(n+s)^2 = (2+o(1))s^2 \leq O\left(\frac{1}{n^2}\right) 2^{2n}$$

Platí tedy také

$$((n+s)+2(n+s)^2)^s \leq o(1)2^{2ns} = o\left(2^{2^n}\right) \ .$$

Funkcí složitosti nejvýše s je tedy méně než všech funkcí a z toho již plyne tvrzení věty. □

5 Lineární funkce a multilineární polynomy

Dokažme nejprve dvě charakterizace lineárních funkcí, které jsou speciálním případem vlastnosti lineární funkce $f(u)$ na obecným tělesem, že pro každý vektor w je $f(u+w)-f(u)$ jako funkce u konstantní.

Věta 5.1 *Booleovská funkce f je lineární právě tehdy, když pro libovolný index $i = 1, \ldots, n$ je funkce $f(x \oplus e_i) \oplus f(x)$ konstantní, tedy nezávisí na ohodnocení proměnných x.*

Poznamenejme, že vlastnost uvedená ve větě znamená, že hodnota funkce závisí na proměnné x_i buď pro každé ohodnocení ostatních proměnných nebo naopak pro žádné.

Důkaz. Jestliže $f(x) = \bigoplus_{i=1}^n a_i x_i \oplus b$, pak pro každé x platí $f(x \oplus e_i) \oplus f(x) = a_i$, což je konstanta.

Pro důkaz opačného směru předpokládejme, že $f(x \oplus e_1) \oplus f(x)$ je konstanta, kterou označíme a_1, a dokažme, že pro libovolné ohodnocení proměnných $x = (x_1, \ldots, x_n)$ platí

$$f(x) = a_1 x_1 \oplus f(0, x_2, \ldots, x_n) \ . \tag{5}$$

Tato identita je ekvivalentní tomu, že

$$a_1 x_1 = f(x_1, x_2, \ldots, x_n) \oplus f(0, x_2, \ldots, x_n) \ .$$

Tento vztah platí pro $x_1 = 0$ triviálně a pro $x_1 = 1$ vyplývá z předpokladu o funkci f. Indukcí podle počtu proměnných dostaneme z (5), že pro vhodné konstanty a_i, $i = 1, \ldots, n$ platí identita

$$f(x) = a_1 x_1 \oplus \ldots \oplus a_n x_n \oplus f(0^n) \ .$$

Funkce f je tedy lineární, což jsme měli ukázat. □

Pro úplnost ukažme ještě obecnější identitu, která také charakterizuje lineární funkce.

Věta 5.2 *Booleovská funkce f je lineární právě tehdy, když pro libovolné tři vektory $u, v, w \in \{0,1\}^n$ platí*

$$f(u) \oplus f(u \oplus w) = f(v) \oplus f(v \oplus w) \ . \tag{6}$$

Důkaz. Jestliže $f(x) = \bigoplus_{i=1}^{n} a_i x_i \oplus b$, pak $f(u \oplus w) \oplus f(u) = \bigoplus_{i=1}^{n} a_i w_i$ a také $f(v \oplus w) \oplus f(v) = \bigoplus_{i=1}^{n} a_i w_i$. Z toho plyne, že lineární funkce splňují identitu (6).

Pro důkaz opačného směru předpokládejme, že funkce f splňuje (6). Pak také splňuje podmínku z Věty 5.1, která je speciálním případem (6) pro $w = e_i$, a funkce f je tedy lineární. □

Polynom nazveme multilineární, pokud v něm má každá proměnná stupeň nejvýše 1. Multilineární polynom nad dvouprvkovým tělesem je tedy polynom tvaru

$$\bigoplus_{I \subseteq \{1,\ldots,n\}} a_I \prod_{i \in I} x_i \ ,$$

kde $a_I \in \{0,1\}$.

Věta 5.3 *Pro každou funkci daných proměnných existuje právě jeden multilineární polynom nad dvouprvkovým tělesem, který ji vyjadřuje.*

Důkaz. Pokud je funkce rovna 1 právě na jednom bodě krychle, označme jej a, pak ji lze vyjádřit polynomem

$$\delta_a(x) = \prod_{i=1}^{n} (x_i \oplus a_i \oplus 1) \ .$$

Obecnou funkci pak vyjádříme ve tvaru

$$f(x) = \bigoplus_{f(a)=1} \delta_a(x) \ .$$

Abychom ukázali, že polynom vyjadřující danou funkci je jednoznačně určen, dokažme, že počet různých polynomů n proměnných je roven počtu všech funkcí

n proměnných. To plyne z toho, že jak funkce, tak polynom, je určen právě 2^n parametry s hodnotami v $\{0,1\}$ a obě množiny tedy mají mohutnost právě 2^{2^n}. □

Důsledkem jednoznačnosti multilineárního polynomu pro Booleovskou funkci je například to, že funkce, kterou lze vyjádřit nelineárním polynomem není lineární, protože v opačném případě by byla vyjádřena ještě jiným polynomem, což by byl spor s jednoznačností. Konjunkce $x \wedge y = xy$ a disjunkce $x \vee y = x \oplus y \oplus xy$ tedy nejsou lineární.

6 Úplnost bází

Připoměňme, že báze se nazývá úplná, pokud formule složené z proměnných a ze spojek v dané bázi umožňují vyjádřit libovolnou Booleovskou funkci. Dříve uvedená báze $\{0, 1, \neg, \vee, \wedge\}$ je úplná, jak plyne například z Věty 1.2 nebo z Věty 2.1. Protože $\neg x = x \oplus 1$ a $x \vee y = (x \oplus 1)(y \oplus 1) \oplus 1$, je úplná i báze $\{0, 1, \oplus, \wedge\}$. Existují také dvě jednoprvkové úplné báze, konkrétně {NAND} a {NOR}, obsahující negovanou binární konjunkci nebo negovanou binární disjunkci. Například pro spojku NAND dostaneme postupně $\neg x = x$ NAND x, $1 = x$ NAND $(\neg x)$, $0 = \neg 1$, $x \wedge y = \neg(x$ NAND $y)$ a $x \vee y = (\neg x)$ NAND $(\neg y)$. Podobně lze ověřit, že báze $\{0, \Rightarrow\}$ je úplná.

Důležitým příkladem neúplné báze je báze $\{0, 1, \vee, \wedge\}$ vyjadřující pouze monotonní funkce. Z Věty 1.5 plyne, že umožňuje vyjádřit všechny monotonní funkce.

Další příklady bází, které nejsou úplné, dostaneme pomocí následujících tříd Booleovských funkcí, které jsou uzavřeny na skládání a neobsahují všechny Booleovské funkce. Řekneme, že Booleovská funkce je

- typu T_0, pokud $f(0, \ldots, 0) = 0$.
- typu T_1, pokud $f(1, \ldots, 1) = 1$.
- samoduální (S), pokud $f^* = f$, tj. $f(\neg x) = \neg f(x)$.
- monotonní (M), pokud $x \leq y$ implikuje $f(x) \leq f(y)$.
- lineární (L), pokud je tvaru $f(x) = a_1 x_1 \oplus \ldots \oplus a_n x_n \oplus b$, kde $\oplus$ je sčítání modulo 2.

V následující tabulce jsou příklady bází, které generují jednotlivé třídy.

T_0	$\{\oplus, \wedge\}$
T_1	$\{\Rightarrow, \wedge\}$
S	$\{\neg x, xy \vee xz \vee yz\}$
M	$\{0, 1, \vee, \wedge\}$
L	$\{1, \oplus\}$

Je zřejmé, že pokud všechny spojky báze patří do některé z výše uvedených pěti tříd, pak daná báze vyjařuje jen funkce z této třídy a není tedy úplná. Ukažme, že platí i opačné tvrzení, a sice, že každá báze, která není úplná, je podmnožinou některé z uvedených tříd. Platí tedy následující charakterizace.

Věta 6.1 *Báze spojek je úplná právě tehdy, když není obsažena v žádné ze tříd T_0, T_1, S, M a L.*

Důkaz. Pokud je báze obsažena v některé z výše uvedených tříd, pak generuje pouze funkce z této třídy a není úplná. Pokud není obsažena v žádné z uvedených tříd, pak dokážeme, že je úplná. Nechť $\alpha_0, \alpha_1, \beta, \gamma, \delta$ jsou spojky z báze, přičemž α_k není z T_k, β není samoduální, γ není monotonní a δ není lineární. Množina funkcí $\{\alpha_0(x, x, \ldots, x), \alpha_1(x, x, \ldots, x)\}$ obsahuje buď funkci $\neg x$ nebo obě konstanty $0, 1$. Další krok konstrukce závisí na tom, který z těchto dvou případů nastává.

V prvním případě lze spojkami báze vyjádřit funkci $\neg x$. Protože β není samoduální, existuje vstup a tak, že $\beta(a) = \beta(\neg a)$. Nechť $l_1, \ldots, l_n$ je posloupnost literálů x a $\neg x$ taková, že $l_i = x$, pokud $a_i = 0$, a $l_i = \neg x$, pokud $a_i = 1$. Pak pro funkci $\varphi(x) = \beta(l_1, \ldots, l_n)$ platí $\varphi(0) = \beta(a) = \beta(\neg a) = \varphi(1)$ a je to tedy konstanta. Tím je dokázáno, že spojkami báze lze vyjádřit některou konstantu. Protože lze vyjádřit i negaci, lze vyjádřit obě konstanty.

Ve druhém případě lze pomocí α vyjádřit obě konstanty. Protože γ není monotonní, existují vstupy a, b takové, že $a \leq b$, $\gamma(a) = 1$, $\gamma(b) = 0$ a existuje pouze jeden index i pro který je $a_i \neq b_i$. Funkce $\gamma(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)$ realizuje $\neg x$ pomocí konstant a γ.

V obou případech tedy můžeme pomocí spojek báze vyjádřit $0, 1, \neg x$. Nyní stačí ukázat, že lze vyjádřit také $x \wedge y$ nebo $x \vee y$. Označme

$$\begin{array}{lcl} a_1 & = & \delta(1, 0, \ldots, 0), \\ a_2 & = & \delta(0, 1, \ldots, 0), \\ \ldots & & \\ a_n & = & \delta(0, 0, \ldots, 1), \\ b & = & \delta(0, 0, \ldots, 0). \end{array}$$

Uvažme funkci

$$\delta' = (a_1 \oplus b)x_1 \oplus (a_2 \oplus b)x_2 \oplus \ldots \oplus (a_n \oplus b)x_n \oplus b.$$

Funkce δ a δ' se shodují na vstupech s nejvýše jednou jedničkou, ale na některém vstupu s větším počtem jedniček se liší, protože δ není a δ' je lineární. Nechť c je některý vstup s minimálním počtem jedniček, pro který je $\delta(c) \neq \delta'(c)$ a nechť j, k jsou indexy některých dvou jedniček v tomto vstupu. Nechť funkce $\varphi(x, y)$, resp. $\varphi'(x, y)$ vznikne z δ, resp. δ' dosazením c_i na pozici i, pokud $i \notin \{j, k\}$, x na pozici j a y na pozici k. Funkce $\varphi'(x, y)$ je lineární a $\varphi(x, y)$ se od ní liší pro

jedinou kombinaci vstupů, konkrétně pro $(x, y) = (1, 1)$. Podle Lemmatu 6.2 je funkce dvou proměnných lineární právě tehdy, když má sudý počet jedniček ve své tabulce. Z toho plyne, že $\varphi(x, y)$ není lineární. Existují právě 4 nelineární funkce dvou proměnných s jednou jedničkou v tabulce. Všechny tyto funkce lze vhodnou kombinací negací u vstupů převést na konjunkci. Nelineární funkce dvou proměných se třemi jedničkami v tabulce lze vhodnou kombinací negací u vstupů převést na disjunkci. V obou případech tedy lze vyjádřit konjunkci nebo disjunkci. Protože tyto dvě funkce lze mezi sebou převést vhodnou kombinací negací vstupů a výstupu, dostaneme v obou případech z φ konjunkci i disjunkci. Protože funkce φ je vyjádřitelná funkcemi ze zkoumané báze, dokázali jsme, že ze spojek zkoumané báze lze vyjádřit $0, 1, \neg x, x \vee y, x \wedge y$. Tím je věta dokázána. □

Lemma 6.2 *Funkce dvou proměnných je lineární právě tehdy, když má sudý počet jedniček ve své tabulce.*

Důkaz. Pokud je funkce $f(x, y)$ lineární, je podle Věty 5.1 $f(0, y) \oplus f(1, y)$ konstanta, což implikuje, že platí

$$f(0,0) \oplus f(1,0) \oplus f(0,1) \oplus f(1,1) = 0 \ , \tag{7}$$

tedy f má sudý počet jedniček v tabulce.

Nechť naopak $f(x, y)$ splňuje (7). Označme jako g funkci

$$g(x, y) = (f(1, 0) \oplus f(0, 0))x \oplus (f(0, 1) \oplus f(0, 0))y \oplus f(0, 0)$$

Tato funkce se shoduje s f na vstupech s nejvýše jednou jedničkou, což lze ověřit výčtem pro všechny takové vstupy, tedy pro $(x, y) = (0, 0), (1, 0), (0, 1)$. Dále dostaneme $g(1, 1) = f(1, 0) \oplus f(0, 1) \oplus f(0, 0)$. Z předpokladu (7) tedy dostáváme, že $g(1, 1) = f(1, 1)$, tedy $g = f$. Protože je g lineární, je i f lineární. □

Existuje celkem 16 B. funkcí 2 proměnných. Tyto funkce roztřídíme do skupin tak, že funkce v jedné skupině lze na sebe převést negací některých vstupů nebo výstupu, což jsou transformace, které zachovávají linearitu. V následující tabulce je pro každou skupinu funkcí uvedeno, kolik jedniček tyto funkce mají ve své tabulce a kolik funkcí do skupiny patří.

skupina funkcí	počet jedniček	počet funkcí
konstanty	0, 4	2
funkce jedné proměnné	2	4
funkce ekvivalentní konjunkci (disjunkci)	1, 3	8
funkce ekvivalentní $\oplus$	2	2

7 Oddělování modelů

Oddělením modelu M_1 od modelu M_2 budeme rozumět to, že $\mathrm{Poly}(M_1) \not\subseteq \mathrm{Poly}(M_2)$. Jinak řečeno, existuje posloupnost funkcí, které mají v M_1 polynomiální reprezentaci, ale všechny reprezentace této posloupnosti v M_2 jsou větší než polynomiální. Oddělení nazveme exponenciální, jestliže dokážeme, že ve druhém modelu mají všechny reprezentace uvažovaných funkcí exponenciální velikost. Obecně připouštíme, že skutečně dokážeme jen existenci oddělující posloupnosti funkcí. Silnější výsledek ovšem je, když lze najít konkrétní oddělující posloupnost.

Pro důkazy oddělení výše popsaných Booleovských modelů zavedeme několik příkladů Booleovských funkcí. Každé oddělení modelů vyžaduje dokázat dolní odhad složitosti nějaké funkce pro některý model. Tato část oddělování je obvykle obtížnější. Proto oddělování rozlišíme podle toho, vůči kterému modelu je dokázán dolní odhad složitosti.

Následující tabulka shrnuje oddělení modelů v diagramu, která dokážeme.

Model	Oddělen od	funkce
DNF	stromy	free-dnf
CNF	stromy	free-cnf
formule	DNF, CNF	parita
read-once r.d.	DNF, CNF	parita
read-once r.d.	OBDD	ISA, shifted-equality, select-row-col
rozhodovací diagramy	read-once r.d.	pointrové funkce, plane, půl-kliky
DNF	read-once r.d.	plane
CNF	read-once r.d.	plane*

7.1 Vyjádření parity

Funkcí par_n budeme rozumět funkci proměnných $x_1, \ldots, x_n$ vyjádřenou jako $\mathrm{par}_n(x) = \bigoplus_{i=1}^{n} x_i$.

Věta 7.1 *Velikost formule v bázi* $\{\neg, \vee, \wedge\}$ *pro funkci* par_n *je* $O(n^2)$.

Důkaz. Vytvoříme nejprve pro par_n formuli hloubky $\log n + O(1)$ v bázi $\oplus$. Pak každý výskyt $\oplus$ nahradíme ekvivalentním vyjádřením pomocí $\{\vee, \wedge, \neg\}$, což si vynutí vytvoření identických kopií některých částí formule. Odhad složitosti výsledné formule dostaneme tak, že zjistíme hloubku nové formule, do níž ale budeme započítávat jen binární spojky. Protože hloubka nové formule je $2 \log n + O(1)$, je její velikost $O(n^2)$. □

Věta 7.2 *Velikost read-once rozhodovacího diagramu pro funkci* par_n *je* $O(n)$.

Důkaz. Rozhodovací diagram pro par_n vytvoříme jako matici uzlů rozměru $(n+1) \times 2$. Pro $i = 1, \ldots, n$ testují oba vrcholy v i-tém řádku proměnnou x_i. Je-li její hodnota 0, přechází výpočet svisle dolů. Je-li hodnota 1, přechází výpočet

na další řádek, ale do opačného sloupce. Počáteční uzel je na řádku 1 vlevo. Na řádku $n+1$ jsou koncové uzly s hodnotami 0 a 1. □

7.2 Dolní odhady pro disjunktivní normální formu

Věta 7.3 *Jestliže všechny primární implikanty funkce f mají délku aspoň k, pak*

$$\mathrm{dnf}(f) \geq \frac{|Supp(f)|}{2^{n-k}}.$$

Důkaz. Jestliže $f = \bigvee_{i=1}^{l} m_i$, pak každý z monomů m_i splňuje $m_i \leq f$, obsahuje nějaký primární implikant f a má tedy délku alespoň k. Navíc,

$$\sum_x f(x) \leq \sum_{i=1}^{l} \sum_x m_i(x).$$

Protože monom délky alespoň k má nejvýše 2^{n-k} splňujících ohodnocení, dostaneme

$$|\mathrm{Supp}(f)| \leq l \cdot 2^{n-k}.$$

Tím je tvrzení věty dokázáno. □

Cvičení. V sekci 1.4 jsme dokázali, že $\mathrm{dnf}(\mathrm{par}) \geq 2^{n-1}$. Ověřte, že tento odhad a odhad $\mathrm{dnf}(\neg\mathrm{par}) \geq 2^{n-1}$ plynou také z Věty 7.3.

Věta 7.4 *Nechť f je funkce n proměnných a pro množinu $A \subseteq \{0,1\}^n$ platí, že každý primární implikant f má nejvýše r splňujících ohodnocení v množině A. Pak*

$$\mathrm{dnf}(f) \geq \frac{|Supp(f \wedge X_A)|}{r},$$

kde X_A je charakteristická funkce množiny A.

Důkaz. Označme $l = \mathrm{dnf}(f)$ a nechť $f = \bigvee_{i=1}^{l} m_i$, kde m_i jsou primární implikanty. Pak také platí $f \wedge X_A = \bigvee_{i=1}^{l} m_i \wedge X_A$, a tedy také

$$\sum_{x \in A} f(x) \leq \sum_{i=1}^{l} \sum_{x \in A} m_i(x).$$

Protože každý sčítanec v pravé straně je nejvýše r, dostaneme

$$|\mathrm{Supp}(f \wedge X_A)| \leq lr.$$

Tím je tvrzení věty dokázáno. □

Cvičení. Pomocí Věty 7.4 dokažte $\mathrm{dnf}(\mathrm{par}_n \vee \bigwedge_{i=1}^{n} \neg x_i) \geq 2^{n-1}$.

Označme jako T_k^n funkci "alespoň k jedniček z n proměnných". Pak platí

Důsledek 7.5 $\mathrm{dnf}(T_k^n) \geq \binom{n}{k}$.

Důkaz. Mintermy funkce T_k^n jsou právě všechny monomy obsahující právě k pozitivních literálů. □

Zaveďme nyní funkce $\mathrm{equality}_n(x, y)$ a $\text{free-cnf}_n(x, y)$ $2n$ proměnných tak, že pro libovolné $x, y \in \{0,1\}^n$ platí

$$\begin{aligned} \mathrm{equality}_n(x,y) &= \bigwedge_{i=1}^{n} x_i = y_i, \\ \text{free-cnf}_n(x,y) &= \bigwedge_{i=1}^{n} (x_i \vee y_i). \\ \text{free-dnf}_n(x,y) &= \bigvee_{i=1}^{n} (x_i \wedge y_i). \end{aligned}$$

Věta 7.6 *Pro každé n platí* $\mathrm{dnf}(\mathrm{equality}_n) \geq 2^n$ *a* $\mathrm{dnf}(\text{free-cnf}_n) \geq 2^n$.

Důkaz. K důkazu první nerovnosti si stačí uvědomit, že funkce $\mathrm{equality}_n$ má 2^n splňujících ohodnocení a každé z nich určuje primární implikant délky $2n$. Tyto implikanty tvoří množinu všech primárních implikantů funkce a tedy podle Věty 7.3 je $\mathrm{dnf}(\mathrm{equality}_n) \geq 2^n$.

K důkazu druhé nerovnosti lze snadno ověřit, že primární implikanty free-cnf_n jsou právě ty částečné vstupy, které pro každé $i = 1, \ldots, n$ splňují $(x_i, y_i) = (1, 0)$ nebo $(x_i, y_i) = (0, 1)$. Takovýchto částečných vstupů je právě 2^n. Protože free-cnf_n je monotonní, dostáváme z Věty 1.7 $\mathrm{dnf}(\text{free-cnf}_n) \geq 2^n$. □

Důsledek 7.7 $\mathrm{cnf}(\text{free-dnf}_n) \geq 2^n$.

Důkaz. Využijeme, že platí $\text{free-dnf}_n = \text{free-cnf}_n^*$ a pro každou f platí $\mathrm{cnf}(f) = \mathrm{dnf}(f^*)$. □

7.3 Dolní odhady pro rozhodovací stromy

Nejjednodušší dolní odhad pro stromy odvodíme z odhadu pro DNF a CNF pomocí následujícího lemmatu.

Lemma 7.8 *Pro každou funkci f platí* $\mathrm{dnf}(f) + \mathrm{cnf}(f) \leq \mathrm{dt}(f)$.

Důkaz. Uvažme některý minimální rozhodovací strom pro funkci f. Každé hraně přiřadíme literál, který je splněn, pokud výpočet prochází přes tuto hranu. Z uzlu, který testuje x_i tedy vychází jedna hrana ohodnocená x_i a jedna ohodnocená $\neg x_i$. Každému listu s hodnotou 1 přiřadíme monom, který je konjunkcí literálů na hranách cesty z kořene do daného listu. Je zřejmé, že tento monom je implikantem f a navíc, disjunkce všech takto získaných monomů je DNF pro f.

Dále uvažme strom pro $\neg f$, který získáme obrácením hodnot v listech. Stejnou konstrukcí jako výše získáme DNF pro $\neg f$ a z ní negací získáme CNF pro f.

Součet velikostí získané DNF a CNF označme s. Tento součet je roven počtu listů ve výchozím stromu. Protože strom byl minimální, je $s = dt(f)$. Na druhé straně, $\mathrm{dnf}(f) + \mathrm{cnf}(f) \leq s$, protože minimální DNF a CNF mají velikost nejvýše rovnu velikosti nalezených DNF a CNF. □

Toto lemma dovoluje automaticky přenést dolní odhady složitosti pro DNF i CNF na stromy. Platí tedy následující odhady pro dříve zavedené funkce $2n$ proměnných.

Důsledek 7.9

$$\begin{aligned} \mathrm{dt}(\mathrm{equality}_n) &\geq 2^n \\ \mathrm{dt}(\text{free-dnf}_n) &\geq 2^n \\ \mathrm{dt}(\text{free-cnf}_n) &\geq 2^n \end{aligned}$$

Pokud ještě využijeme skutečnost, že $\mathrm{dnf}(\text{free-dnf}_n) = n$ a $\mathrm{cnf}(\text{free-cnf}_n) = n$, dostaneme exponenciální rozdíl mezi $\mathrm{dnf}(f)$ a $\mathrm{dt}(f)$ pro $f = \text{free-dnf}_n$ a exponenciální rozdíl mezi $\mathrm{cnf}(f)$ a $\mathrm{dt}(f)$ pro $f = \text{free-cnf}_n$.

Odvodíme ještě poměrně přesný horní i dolní odhad velikosti rozhodovacího stromu pomocí jednoduché indukce.

Věta 7.10 *Pro každé n platí* $\mathrm{dt}(\mathrm{equality}_n) = \Theta(2^n)$ *a* $\mathrm{dt}(\text{free-cnf}_n) = \Theta(2^n)$.

Důkaz. Označme nejprve $s(n) = \mathrm{dt}(\text{free-cnf}_n)$. Rozhodovací strom pro $\text{free-cnf}_n(x_1, y_1, \ldots, x_n, y_n)$ lze zkonstruovat tak, že nejprve otestuje x_1. Pokud $x_1 = 1$, pak počítá funkci $\text{free-cnf}_{n-1}(x_2, y_2, \ldots, x_n, y_n)$, v případě $x_1 = 0$ počítá $y_1 \wedge \text{free-cnf}_{n-1}(x_2, y_2, \ldots, x_n, y_n)$. Z této konstrukce plyne $s(n) \leq 2s(n-1) + 1$. Spolu s $s(1) = 3$ to dává $\mathrm{dt}(\text{free-cnf}_n) = s(n) \leq 2^{n+1} - 1$.

Pro důkaz dolního odhadu $s(n)$ uvažme libovolný strom pro free-cnf_n. Bez újmy na obecnosti předpokládejme, že tento strom testuje v kořeni proměnnou x_1. Pokud v podstromu pro $x_1 = 0$ dosadíme $y_1 = 1$, pak tento podstrom počítá funkci

$$\text{free-cnf}_{n-1}(x_2, y_2, \ldots, x_n, y_n)$$

a má tedy velikost alespoň $s(n-1)$. Podstrom pro $x_i = 1$ počítá tutéž funkci pro libovolné dosazení za y_i a má tedy také velikost alespoň $s(n-1)$. Celkem jsme tedy získali $s(n) \geq 2s(n-1)$. Spolu s $s(1) = 3$ to dává $s(n) \geq 3 \cdot 2^{n-1}$.

Jestliže označíme $s'(n) = \mathrm{dt}(\mathrm{equality}_n)$, pak podobným způsobem jako výše lze dokázat $s'(n) \leq 2s'(n-1) + 2$, $s'(n) \geq 2s'(n-1)$ a $s'(1) = 4$. Z toho plyne požadovaný odhad $\mathrm{dt}(\mathrm{equality}_n) = \Theta(2^n)$. □

Cvičení. Dokažte, že $\mathrm{dt}(T_k^n) = \binom{n+1}{k}$.

7.4 Dolní odhady pro read-once r.d.

Jako základní prostředek použijeme tzv. k-mixed vlastnost. Tato vlastnost dává pro některé funkce velmi dobrý odhad poměrně jednoduchým způsobem.

Definice 7.11 Funkci f nazveme k-mixed, jestliže pro každou množinu $A \subseteq X$ velikosti $|A| = k$ a pro libovolné dva různé částečné vstupy $a, b \in \{0,1\}^A$ existuje $c \in \{0,1\}^{X \setminus A}$ tak, že platí $f(a,c) \neq f(b,c)$.

Věta 7.12 *Jestliže f je k-mixed, pak* $1\text{-bdd}(f) \geq 2^k - 3$.

Důkaz. Ukážeme, že v každém read-once rozhodovacím diagramu pro f platí, že žádné dvě cesty z počátečního uzlu délky nejvýše $k-1$ nevedou do téhož uzlu. Z toho plyne, že uvažovaný read-once diagram je strom až do hloubky $k-1$ a má tedy alespoň $2^k - 1$ uzlů.

Důkaz provedeme sporem. Nechť cesty α a β délky nejvýše $k-1$ vedou do téhož uzlu v. Nechť A resp. B je množina proměnných testovaných na cestě α resp. β.

Pokud cesty α a β testují stejnou množinu proměnných, pak tyto cesty definují dva různé částečné vstupy a, b na proměnných z A, jejichž výpočty se sejdou v uzlu v. Protože uvažujeme read-once diagram, žádná z proměnných z A již v dalším výpočtu z uzlu v nemůže být testována a výpočty navazující na α i β proběhnou stejně.. Označme jako c libovolné dosazení za dosud neurčené proměnné. Pak platí $f(a,c) = f(b,c)$ a funkce tedy není $|A|$-mixed ani k-mixed, protože $|A| \leq k$.

Pokud cesty α a β testují různé množiny proměnných, zvolme proměnnou x_i, která je bez újmy na obecnosti testována v β a nikoli v α. Nechť cesta α a $x_i = 0$ definují vstup a a cesta α a $x_i = 1$ vstup b, oba na stejné množině $|A| + 1 \leq k$ proměnných. Vstupy a, b jsou různé a jejich výpočty se sejdou ve v. Nechť c je libovolné doplnění hodnot proměnných neurčených v a a b. Pak platí $f(a,c) = f(b,c)$ a funkce tedy není k-mixed. □

Nechť p je prvočíslo. Na prvcích $Z_p^3 \setminus \{[0,0,0]\}$ zavedeme ekvivalenci. Prvky $[x_1, x_2, x_3]$ a $[y_1, y_2, y_3]$ budou ekvivalentní právě tehdy, když existuje $\alpha \neq 0$ tak, že $[x_1, x_2, x_3] = [\alpha y_1, \alpha y_2, \alpha y_3]$. Takto získáme $(p^3 - 1)/(p - 1) = p^2 + p + 1$ bodů projektivní roviny. Přímky získáme stejným způsobem, ale budeme je značit $[a_1, a_2, a_3]$. Přímka $[a_1, a_2, a_3]$ obsahuje bod $[x_1, x_2, x_3]$ právě tehdy, když $a_1x_1 + a_2x_2 + a_3x_3 = 0$. Netriviální rovnice tohoto tvaru má v Z_p právě p^2 řešení. Vypustíme-li nulové řešení, má taková rovnice právě $(p^2 - 1)/(p - 1) = p + 1$

neekvivalentních řešení. Každá přímka tedy obsahuje $p+1$ bodů a každý bod je obsažen v $p+1$ přímkách. Jednoduchým rozborem případů lze dokázat, že každé dvě různé přímky se protínají právě v jednom bodě a že každé dva různé body určují právě jednu přímku.

Množinu bodů budeme značit $B(p)$ a množinu přímek $L(p)$. Pro libovolné p pak zkonstruujeme Booleovskou funkci $\mathrm{plane}_p(x)$ o p^2+p+1 proměnných, které odpovídají bodům z $B(p)$. Funkce je vyjádřena dnf, jejíž monomy odpovídají přímkám. Přímky považujeme za množiny bodů.

$$\mathrm{plane}_p(x) = \bigvee_{\ell \in L(p)} \bigwedge_{j \in \ell} x_j.$$

Počet proměnných funkce $\mathrm{plane}_p(x)$ je $n = p^2+p+1$. Lze ověřit, že platí $p = \sqrt{n-3/4} - 1/2$. Dokážeme následující větu.

Věta 7.13 *Pro každé dost velké prvočíslo p platí* $1\text{-bdd}(\mathrm{plane}_p) \geq 2^p - 3 = \Omega\left(2^{\sqrt{n}}\right)$.

Důkaz. Zvolme prvočíslo p a dokažme, že funkce $\mathrm{plane}_p(x)$ je p-mixed. Nechť A je libovolná množina p proměnných. O jejích prvcích budeme mluvit jako o bodech. Protože $|A| < p+1$, neobsahuje A žádnou přímku. Zvolme libovolně dvě různá dosazení a, b za proměnné z A a nechť $x_j \in A$ je některá z proměnných, ve kterých se a a b liší. Protože bodem x_j prochází $p+1$ přímek, existuje přímka ℓ tak, že $A \cap \ell = \{x_j\}$.

Tvrdíme, že žádná přímka ℓ' kromě ℓ není obsažena celá v $A \cup \ell$. Kdyby taková přímka ℓ' existovala, pak $|\ell' \setminus \ell| = p$ a $\ell' \setminus \ell \subseteq A \setminus \ell$. To je ve sporu s tím, že $|A \setminus \ell| = p-1$.

Nechť c určuje dosazení 0 za všechny proměnné mimo $A \cup \ell$ a 1 za proměnné v $\ell \setminus A$. Dosazení c zaručuje, že všechny monomy v definici plane_p jsou vynulovány, kromě monomu, který odpovídá přímce ℓ. Tento monom pak bude roven x_j. Dostaneme tedy $\mathrm{plane}_p(a,c) \neq \mathrm{plane}_p(b,c)$.

Dokázali jsme, že funkce plane_p je p-mixed. Z toho plyne odhad složitosti $2^p - 3$. Protože $p = \sqrt{n-3/4} - 1/2 = \sqrt{n} + O(1)$, plyne z toho i druhý odhad ve znění věty. □

Z dokázané věty plyne nepolynomiální dolní odhad složitosti read-once rozhodovacích diagramů pro funkci plane_p. Na druhé straně, tato funkce je z definice vyjádřitelná pomocí DNF velikosti $n = p^2+p+1$, protože toto je počet přímek v uvažované geometrii. Pokud ještě vezmeme v úvahu, že funkce parita má read-once rozhodovací diagramy lineární velikosti (Věta 7.2), zatímco DNF pouze exponenciální velikosti (Věta 1.4), dostaneme, že modely DNF a read-once rozhodovací diagramy jsou z hlediska vyjadřovací síly navzájem neporovnatelné.

Ukážeme ještě další příklad funkce, která je k-mixed. Budeme uvažovat grafy na $m = 2k+1$ vrcholech. Každé neuspořádané dvojici vrcholů přiřadíme proměnnou. Libovolné ohodnocení těchto $\binom{m}{2}$ proměnných budeme interpretovat jako kód grafu, který má hrany mezi těmi dvojicemi vrcholů, kterým odpovídá proměnná s hodnotou 1. Definujme funkci par-triang$_k$ tak, že par-triang$_k(x) = 1$, pokud graf kódovaný pomocí x obsahuje lichý počet trojúhelníků. Bez důkazu uveďme následující tvrzení.

Věta 7.14 *Pro každé $k \geq 1$ je funkce par-triang$_k$ k-mixed.*

Protože počet proměnných funkce par-triang$_k$ je $n = \binom{2k+1}{2} = (2 + o(1))k^2$, dostaneme pro tuto funkci dolní odhad složitosti read-once rozhodovacích diagramů $2^k - 3 = 2^{(1/2+o(1))\sqrt{n}}$. Toto ale není nejlepší známý odhad pro tuto funkci. Jiným způsobem lze dokázat dolní odhad složitosti $\Omega\left(2^{n/48}\right)$.

7.5 Formule a read-once diagramy

Pokud budeme měřit složitost plane$_p$ pomocí míry pro obecné formule, tedy jako počet výskytů proměnných, je třeba vzít v úvahu, že délka všech monomů v DNF pro tuto funkci je $p+1$, tedy složitost příslušné formule je $n(\sqrt{n} + O(1)) = (1+o(1))n^{3/2}$, což je shora omezeno polynomem. Obecné Booleovské formule tedy nelze vnořit do read-once rozhodovacích diagramů. Není známo, zda platí opačná inkluze. Platí ale následující tvrzení, které ukazuje, že read-once rozhodovací diagramy lze vnořit do obecných formulí právě tehdy, když lze obecné rozhodovací diagramy vnořit do obecných formulí.

Věta 7.15 *Kdyby existoval polynomiální převod read-once rozhodovacích diagramů na formule, pak by existoval i polynomiální převod obecných rozhodovacích diagramů na formule.*

Důkaz. Vezměme obecný rozhodovací diagram obsahující s větvících uzlů. Zvolme s nových proměnných a test v každém uzlu nahraďme testem jiné proměnné z těchto nových proměnných. Na získaný diagram použijme předpokládaný polynomiální převod na formuli a v získané formuli nahraďme zpět každou novou proměnnou odpovídající původní proměnnou. □

7.6 Asymptoticky dobré dolní odhady pro read-once r.d.

V tomto odstavci budeme proměnné Booleovské funkce indexovat od nuly, tedy budeme uvažovat funkce proměnných $x_0, \ldots, x_{n-1}$. Ukážeme dolní odhady pro dvě pointerové funkce, což jsou funkce tvaru $f(x) = x_{\phi(x)}$, kde $\phi : \{0,1\}^n \to \{0, \ldots, m-1\}$, kde m je přirozené číslo a používáme konvenci, že v případě, že $\phi(x) \geq n$, pak $f(x) = x_{\phi(x)} = 0$. Omezíme se na funkce $\phi(x)$ ve tvaru $\phi(x) =$

$\left(\sum_{i=0}^{n-1} w_i x_i\right) \bmod m$, kde $n \leq m \leq 2n$. Koeficienty w_i je třeba zvolit tak, aby v následujícím lemmatu mohlo být r co největší. Ve dvou konkrétních příkladech, které uvedeme, bude $r = n/3$ a $r = n - O(\sqrt{n})$.

Lemma 7.16 *Nechť koeficienty w_i jsou takové, že pro libovolné $i = 0, \ldots, n-1$ a libovolný částečný vstup a fixující nejvýše r proměnných existuje doplnění a na úplný vstup a' tak, že $\phi(a') = i$. Pak funkce $f(x) = x_{\phi(x)}$ je $(r-2)$-mixed.*

Důkaz. Uvažme dva vstupy a, b fixující stejnou množinu $r-2$ proměnných a označme jako $\Delta \in Z_n$ rozdíl příspěvků částečných vstupů a, b do sumy $\phi(x)$, kde bereme v úvahu jen členy pro ohodnocené proměnné. Je-li c libovolné ohodnocení zbylých $n - r + 2$ proměnných, pak $\phi(b, c) - \phi(a, c) = \Delta \bmod m$. Pokud $\Delta = 0$, zvolíme c tak, aby $\phi(a, c) = \phi(b, c)$ byla hodnota indexu, kde se a a b liší. Pak platí $f(a, c) \neq f(b, c)$, a funkce je $(r-2)$-mixed. Je-li $\Delta \neq 0$, nechť i je index libovolné proměnné, která není fixována v a a b a nechť $j = (i + \Delta) \bmod m$. Pokud je $j \geq n$, zvolíme $c_i = 1$. Pokud je $j \leq n$ a proměnná x_j je fixována, zvolíme hodnotu c_i tak, aby se lišila od b_j. Pokud $j \leq n$ a x_j není fixována, zvolíme c_j libovolně a c_i zvolíme různé od c_j. Protože je dosud fixováno nejvýše r proměnných, můžeme zvolit zbývající část ohodnocení c tak, aby $\phi(a, c) = i$ a tedy $\phi(b, c) = j$. Volba ohodnocení c pak zaručuje, že $f(a, c) \neq f(b, c)$ a funkce je tedy $(r-2)$-mixed. □

Nechť n je dělitelné 3. Definujme

$$\phi(x) = \left(\sum_{i=0}^{2/3 \cdot n - 1} 2\, x_i + \sum_{i=2/3 \cdot n}^{n-1} x_i \right) \bmod n.$$

Pak platí následující věta.

Věta 7.17 $\text{1-bdd}(x_{\phi(x)}) \geq 2^{n/3-3} - 1$.

Důkaz. Ukážeme, že lemma 7.16 je splněno pro $r = n/3 - 1$.

Uvažme libovolné i, $0 \leq i \leq n-1$ a libovolný částečný vstup a', který fixuje nejvýše $n/3-1$ proměnných. Zbývá alespoň $2/3 \cdot n + 1$ nefixovaných proměnných, mezi kterými je alespoň jedna s vahou 1. Budeme uvažovat obě možné fixace této proměnné. Získaný částečný vstup budeme označovat $(a, 0)$ nebo $(a, 1)$ podle hodnoty, na kterou byla uvažovaná proměnná fixována. Zbývá alespoň $2/3 \cdot n$ nefixovaných proměnných, mezi kterými je alespoň $n/3$ proměnných s vahou 2. Součet vah u dosud nefixovaných proměnných je tedy alespoň n.

Označme jako j součet příspěvků proměnných fixovaných v $(a, 0)$ do sumy $\phi(x)$. K tomu, aby $\phi(a, 0, b) = i$, kde b je částečný vstup doplňující $(a, 0)$ na úplný vstup, je nutné a stačí, aby součet příspěvků proměnných fixovaných v b byl $(i - j) \bmod n$. Hodnoty proměnných v b budeme volit tak, nejprve všechny

nastavíme na 0 a pak v libovolném pořadí postupně měníme jejich hodnoty na 1 tak dlouho, dokud součet jejich příspěvků do sumy $\phi(x)$ je nejvýše $(i-j)$ mod n. Protože součet vah u proměnných fixovaných v b je alespoň n a změnou jedné proměnné se příspěvek zvýší nejvýše o 2, získáme tak b s příspěvkem $(i-j)$ mod n nebo $(i-j)$ mod $n-1$. Pokud nastala první možnost, pak $\phi(a,0,b)=i$, pokud nastala druhá možnost, pak $\phi(a,1,b)=i$. □

Odhad složitosti z Věty 7.17 lze zlepšit nahrazením funkce $\phi(x)$ jinou funkcí. Označme jako $p(n)$ nejmenší prvočíslo, které splňuje $p(n) \geq n$ a zvolme $\phi'(x) = \left(\sum_{i=0}^{n-1} i\, x_i\right) \bmod p(n)$. K důkazu dolního odhadu složitosti funkce $x_{\phi'(x)}$ budeme potřebovat následující větu z teorie čísel, kterou uvádíme bez důkazu.

Věta 7.18 *Nechť p je prvočíslo, nechť $0 \leq t \leq s \leq p$ a nechť $A \subseteq Z_p$ tak, že $|A| = s$. Nechť B je množina všech prvků Z_p, které lze vyjádřit jako součet prvků některé t prvkové podmnožiny A. Pak $|B| \geq \min(n, t(s-t)+1)$.*

Lze snadno ověřit, že pro libovolné $0 \leq t \leq s$ nastává v odhadu ve větě rovnost, pokud je A interval, např. $A = \{0, 1, \ldots, s-1\}$.

Věta 7.19 *Pro každé dostatečně velké n platí* $1\text{-bdd}(x_{\phi'(x)}) \geq 2^{n-2\sqrt{n+o(n)}}$.

Důkaz. Označme $t = \left\lceil \sqrt{p(n)-1} \right\rceil$ a $s = 2t$. Dokážeme, že $x_{\phi'(x)}$ splňuje Lemma 7.16 pro $r = n-s$. Fixujeme-li $r = n-s$ proměnných, pak zbývá s nefixovaných proměnných. Množinu vah u těchto proměnných označme A. Uvažujeme-li dosazení za tyto proměnné, která fixují t nul a t jedniček, jsou příspěvky těchto proměnných do sumy $\phi'(x)$ právě součty některých t prvků z A, jejíž velikost je s. Podle Věty 7.18 je množina takto dosažitelných součtů rovna množině všech zbytků modulo $p(n)$, protože $t(s-t)+1 \geq p(n)$. Je tedy splněn předpoklad Lemmatu 7.16 pro $r = n-s$ a funkce $x_{\phi'(x)}$ je $(r-2)$-mixed. Z toho dostaneme dolní odhad $1\text{-bdd}(x_{\phi'(x)}) \geq 2^{r-2} - 3 = 2^{n-2t-2} - 3 \geq 2^{n-2\sqrt{p(n)}+O(1)}$. Z výsledků teorie čísel je známo, že $p(n) = n + o(n)$ a tedy $\sqrt{p(n)} + O(1) = \sqrt{n + o(n) + O(\sqrt{n})}$. Tím je odhad z tvrzení věty dokázán. □

7.7 Dolní odhady složitosti pro OBDD

Velikostí OBDD rozumíme počet všech jeho uzlů, včetně výstupních. Pro danou funkci f budeme π-OBDD(f) rozumět velikost nejmenšího π-OBDD pro f a OBDD(f) bude minimum π-OBDD(f) přes všechna π.

Částečný vstup budeme chápat jako dosazení konstant za proměnné ve funkci. Toto dosazení lze chápat dvěma různými způsoby, které se nazývají restrikce na podkrychli a subfunkce. První způsob odpovídá obvyklému významu restrikce

funkce na podmnožinu definičního oboru. Druhý způsob odpovídá substituci konstant za proměnné ve výrazu, který i po substituci formálně považujeme za reprezentaci funkce všech původních proměných. Při zkoumání OBDD budeme používat pojem subfunkce.

Subfunkci funkce f určené částečným vstupem a budeme značit $f|_a$. Formálně považujeme za množinu proměnných subfunkce $f|_a$ množinu všech proměnných funkce f, i když $f|_a$ může podstatně záviset jen na proměnných, které nejsou v a fixovány. Důvodem pro tuto konvenci je, že nechceme rozlišovat subfunkce, které se liší pouze množinou proměných, na kterých funkce podstatně nezávisí. Například, nechť $f = x_1x_2x_3 \vee x_4x_5x_6$ a částečné vstupy a, b reprezentují fixace $a = (x_1 = 0)$ a $b = (x_2 = 0, x_3 = 0)$. Pak platí $f|_a = f|_b$, přestože restrikce f na odpovídající podkrychle jsou funkce různých množin proměnných.

Řekneme, že částečný vstup je konzistentní s uspořádáním π, pokud množina fixovaných proměnných je počáteční úsek π.

Definice 7.20 Výpočtem π-OBDD pro částečný vstup a konzistentní s π nazveme výpočet, který začne v počátečním uzlu a skončí, když dojde do uzlu, ve kterém je testována proměnná nefixovaná v a nebo který je koncový.

Subfunkce a výpočty pro částečené vstupy spolu souvisí následovně.

Lemma 7.21 *Uvažme libovolné* π-OBDD *pro* f. *Pro libovolný částečný vstup* a *konzistentní s* π *platí, že uzel, kde končí výpočet pro částečný vstup* a, *počítá funkci* $f|_a$.

Definice 7.22 Pro libovolnou f a uspořádání π nechť

$$S(f, \pi) = \{f|_a;\ a \text{ je částečný vstup konzistentní s } \pi\}$$

Věta 7.23 *Pro každé* π-OBDD *pro* f, *jehož všechny uzly jsou dosažitelné ze vstupního uzlu, je množina subfunkcí počítaných v jeho uzlech rovna* $S(f, \pi)$.

Důkaz. Ke každému dosažitelnému uzlu v OBDD lze najít částečný vstup a, jehož výpočet v daném uzlu končí. Funkce počítaná v tomto uzlu je tedy $f|_a \in S(f, \pi)$. Opačná inkluze plyne z Lemmatu 7.21. □

Věta 7.24 *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.*

Důkaz. Pro každou subfunkci $g \in S(f,\pi)$ existuje částečný vstup a konzistentní s π tak, že $g = f|_a$. Podle Lemmatu 7.21 počítá uzel, kde končí výpočet pro a, funkci $f|_a$. Počet uzlů π-OBDD pro f je tedy alespoň $|S(f,\pi)|$.

Pro důkaz opačného směru vytvořme graf tak, že každé nekonstatntní subfunkci z $S(f,\pi)$ přiřadíme uzel na hladině $i-1$, pokud x_i je proměnná s nejmenším indexem, na které daná subfunkce podstatně závisí. Tento uzel bude testovat proměnou x_i a hrany z něj povedou do uzlů odpovídajících subfunkcím získaných fixací $x_i = 0$ a $x_i = 1$. Počáteční uzel diagramu bude uzel, který odpovídá funkci f. Uzly odpovídající konstantním subfunkcím budou koncové uzly diagramu. Lze ukázat, že takto získaná struktura je π-OBDD pro f. Protože jeho velikost je $|S(f,\pi)|$, je charakterizace π-OBDD(f) pomocí počtu subfunkcí dokázána.

Pokud má libovolné π-OBDD pro f velikost $|S(f,\pi)|$, musí každé dva jeho uzly počítat jinou subfunkci. Každý uzel musí testovat proměnnou, která má minimální index mezi těmi, na kterých jemu příslušná subfunkce podstatně závisí. Hrany pak vedou do uzlů, které počítají subfunkce vzniklé fixováním testované proměnné a tyto uzly jsou jednoznačně určeny. Celá struktura je tedy izomorfní π-OBDD definovanému na množině subfunkcí f a je tedy jednoznačně určena. □

Dolní odhady složitosti OBDD, které odvodíme v tomto odstavci, budou založeny na odhadech počtu subfunkcí, ale často použijeme pouze jeden řez v daném uspořádání v následujícím smyslu. Řezem pro uspořádání π budeme rozumět rozklad množiny proměnných $X = \{x_1, \ldots, x_n\}$ na množiny A, B tak, že v uspořádání π jsou všechny proměnné z A před všemi proměnnými z B. Označme jako $S(f,A,B)$ podmnožinu $S(f,\pi)$, která je pro řez A,B definována jako

$$S(f,A,B) = \{f|_a;\ a \text{ je libovolná fixace proměnných } A\}$$

Funkce v $S(f,A,B)$ závisí pouze na proměnných z B a jejich počet vyjadřuje množství informace o hodnotách proměnných v A, které je potřeba doplnit k proměnným v B pro vyhodnocení funkce. Tento počet se někdy nazývá šířka (width) OBDD na řezu A,B. Pro jednoduchost zaveďme značení

Definice 7.25 $w_{A,B}(f) = |S(f,A,B)|$.

Z definice $S(f,A,B)$ plyne $|S(f,\pi)| \geq |S(f,A,B)| = w_{A,B}(f)$. Spolu s Větou 7.24 tedy dostáváme

Důsledek 7.26 *Pro libovolnou funkci f a řez A,B v uspořádání π platí* π-OBDD(f) $\geq w_{A,B}(f)$.

Příklad. Pro libovolné k a $m = 2^k$ označme jako $\mathrm{DSA}_m(x,y)$ (direct storage access) funkci $n = k + m$ proměnných, která má k proměnných v bloku x a m proměnných indexovaných od nuly v bloku y a jejíž hodnota je rovna y_i, kde i je hodnota binárního čísla určeného bity bloku x. Definujme uspořádání π_1 tak,

že obsahuje nejprve blok proměnných x a pak blok y. Uspořádání π_2 obsahuje naopak nejprve blok y a pak blok x. Dokažte, že π_1-OBDD(DSA$_m$) $= O(m)$ a π_2-OBDD(DSA$_m$) $\geq 2^m = 2^{n-\log n+O(1)}$.

Následují dolní odhady velikosti OBDD pro některé funkce. Nejprve popíšeme funkci ISA$_m$ (indirect storage access), která umožní dokázat separaci OBDD a stromů. V tomto a dalších příkladech budeme písmeny x, y, z označovat vektory proměnných. Proměnné z vektoru x budeme jednoduše nazývat proměnné x a jejich počet budeme značit $|x|$. Analogicky pro y, z. Nechť bin(x) je hodnota čísla, jehož binární zápis je x.

Definice 7.27 Pro libovolné m nechť ISA$_m(x, y)$ je funkce $n = k + m$ proměnných, kde $|x| = k = \lceil \log m \rceil$ a $|y| = m$, která je definována následovně. Rozdělme proměnné y na $\lfloor m/k \rfloor$ souvislých bloků délky k počínaje nejlevější proměnnou a zbytek velikosti menší než k. Bloky očíslujeme od 0. Jestliže bin(x) není číslem žádného bloku, je ISA$_m(x, y) = 0$. V opačném případě nechť b je blok s číslem bin(x). Je-li bin$(b) > m - 1$, je opět ISA$_m(x, y) = 0$. Jinak ISA$_m(x, y) = y_{\text{bin}(b)}$, kde proměnné y číslujeme od 0.

Postupujeme-li přesně podle definice, je k vyhodnocení ISA$_m$ zapotřebí nejvýše $2k + 1$ dotazů na hodnoty proměnných. To znamená, že velikost rozhodovacího stromu je nejvýše $2^{2k+1} = O(m^2)$. Pro různé vstupy čte strom proměnné v různém pořadí. Následující věta ukazuje, že je to pro velikost stromu podstatné, protože při pevně daném pořadí je exponenciálně veliké dokonce i OBDD.

Věta 7.28 *Pro každé dost velké m platí* OBDD(ISA$_m$) $\geq 2^{(1+o(1))m/\log m} = 2^{(1+o(1))n/\log n}$.

Důkaz. Bloků proměnných y je $\lfloor m/k \rfloor$, kde $k = \lceil \log m \rceil$. Zvolme řez, v němž A obsahuje právě $\lfloor m/k \rfloor - 1$ proměnných y. Pak existuje blok v proměnných y, který celý náleží do B. Fixujme x tak, aby bin(x) bylo číslo tohoto bloku. Označme vzniklou funkci jako $f(y)$ a množinu proměnných y, které náležejí do A označme y_A. Pak různá dosazení za proměnné z y_A dávají různé subfunkce funkce f. Protože $|y_A| = \lfloor m/k \rfloor - 1$, je $w_{A,B}(\text{ISA}_m) \geq w_{A,B}(f) \geq 2^{m/k-2} \geq 2^{m/(\log m+1)-2} = 2^{(1+o(1))m/\log m}$. Protože $n = m+k$, je $n = (1+o(1))m$. Platí tedy také $n/\log n = (1 + o(1))m/\log m$, což dokazuje tvrzení věty. □

Definice 7.29 Pro každé m budeme funkcí sh-equality$_m(x, y, z)$ rozumět funkci $n = k + 2m$ proměnných, kde $|x| = k = \lceil \log m \rceil$ a $|y| = |z| = m$, definovanou následovně. Hodnota sh-equality$_m(x, y, z)$ je rovna jedné právě tehdy, když $y = z^i$, kde $i = \text{bin}(x)$ a z^i je cyklický posun z o i pozic vpravo.

Věta 7.30 *Pro každé dost veliké m platí* OBDD(sh-equality$_m$) $\geq 2^{m/4} = 2^{(1+o(1))n/8}$.

Důkaz. Nechť A, B je libovolný řez, ve kterém do A patří právě polovina z proměnných ze sjednocení bloků y a z. Označme jako y_A, z_A, y_B, z_B množiny proměnných y a z, které patří do jednotlivých částí A a B. Z předpokladu na A, B plyne $|y_A| + |z_A| = m$. Navíc platí $|z_A| + |z_B| = |z| = m$. V dalším budeme předpokládat, že $|y_A| \geq |z_A|$. Opačný případ je analogický.

Protože $|y_A| \geq |z_A|$, dostaneme $|y_A| \geq m/2$ a $|z_B| = m - |z_A| \geq m/2$. Každé dvojici proměnných z y_A, z_B přiřaďme posun i, při kterém budou proměnné dané dvojice porovnávány. To znamená, že příslušná proměnná z y_A ve dvojici bude mít v y stejný index jako druhá proměnná z dvojice ve vektoru z^i. Protože je celkem m různých možných posunů, existuje posun, který je přiřazen alespoň $|y_A| \cdot |z_B|/m \geq (m/2)^2/m = m/4$ dvojicím. Fixujme hodnoty proměnných x podle hodnoty nalezeného posunu a fixujme na nulu všechny proměnné z_A a y_B a ty proměnné z y_A a z_B, které nepatří do dvojic odpovídajících vybranému posunu. Množinu indexů volných proměnných z A resp. B označme A' resp. B'. Je zřejmé, že získaná restrikce funkce sh-equality$_m$ je ekvivalentní rovnosti proměnných $y_{A'}$ a $z_{B'}$ ve vhodném pořadí. Získanou restrikci označme $f'(y_{A'}, z_{B'})$. Jestliže a_1, a_2 jsou dvě různá dosazení za proměnné z $y_{A'}$, pak subfunkce $f'(a_1, z_{B'})$ a $f'(a_2, z_{B'})$ jsou různé. Z toho plyne, že $w_{A,B}(\text{sh-equality}_m) \geq w_{A',B'}(f') \geq 2^{m/4}$. Tím je dokázán první odhad v tvrzení věty. Protože $n = 2m + \log m + O(1)$, platí také $n = 2(1 + o(1))m$, což dokazuje druhý odhad v tvrzení věty. □

Pro další příklad uvažujme x jako jednu proměnnou ($|x| = 1$) a y jako vektor m^2 proměnných uspořádaných do matice m krát m. Navíc, nechť funkce $\text{sel}(x, u, v)$ tří proměnných $x, u, v \in \{0, 1\}$ je definována tak, že $\text{sel}(0, u, v) = u$ a $\text{sel}(1, u, v) = v$.

Definice 7.31 Uvažme Booleovské funkce row$_m$, col$_m$ m^2 proměnných definované následovně:

$\text{row}_m(y) = 1$	právě tehdy, když matice y obsahuje alespoň jeden řádek jedniček;
$\text{col}_m(y) = 1$	právě tehdy, když matice y obsahuje alespoň jeden sloupec jedniček;

Funkce, pro které v následující větě dokážeme stejný dolní odhad složitosti pro OBDD, se velmi liší svojí složitostí pro read-once rozhodovací diagramy. Platí, že $\text{1-bdd}(\text{sel}(x, \text{row}_m(y), \text{col}_m(y))) \leq O(m^2)$, zatímco $\text{1-bdd}(\text{row}_m(y) \vee \text{col}_m(y))$ je exponenciální.

Věta 7.32 *Pro každé $m \geq 1$ platí*

$$\text{OBDD}(\text{sel}(x, \text{row}_m(y), \text{col}_m(y))) \geq 2^{\sqrt{m-1}},$$

$$\text{OBDD}(\text{row}_m(y) \vee \text{col}_m(y)) \geq 2^{\sqrt{m-1}}.$$

Důkaz. Důkazy pro obě funkce provedeme současně, s upozorněním na místa, kde se důkazy liší. Nechť π je libovolné uspořádání. Zvolme řez A, B tak, aby množina A obsahovala právě $m-1$ proměnných y. Množinu těchto proměnných označme y_A a považujme ji za množinu pozic v matici m krát m. Kdyby y_A zasahovala do méně než $\sqrt{m-1}$ řádků a méně než $\sqrt{m-1}$ sloupců, platilo by $|y_A| < m-1$, což je ve sporu s volbou A. Předpokládejme nejprve, že y_A zasahuje do alespoň $\sqrt{m-1}$ řádků. V důkazu pro funkci $\mathrm{sel}(x, \mathrm{row}_m(y), \mathrm{col}_m(y))$ v tomto případě fixujme $x = 0$. V důkazu pro funkci $\mathrm{row}_m(y) \vee \mathrm{col}_m(y)$ v tomto případě fixujme libovolný řádek dosud nefixovaných proměnných na 0. V obou případech tím zajistíme, že výsledek funkce závisí pouze na $\mathrm{row}_m(y)$.

V případě, že y_A zasahuje do méně než $\sqrt{m-1}$ řádků, ale do alespoň $\sqrt{m-1}$ sloupců, postupujme analogicky jako v předchozím případě, ale proměnnnou x fixujeme na 1 a místo řádku zafixujeme některý sloupec na 0.

V dalším tedy předpokládáme, že hodnota funkce je po provedených substitucích rovna $\mathrm{row}_m(y)$. Označme jako k počet řádků, do kterých zasahuje množina y_A. Budeme uvažovat fixace proměnných z y_A, které fixují všechny proměnné patřící do téhož řádku stejně. Zřejmě existuje právě 2^k takových dosazení. Tvrdíme, že libovolná dvě různá dosazení popsaného typu dávají různé subfunkce na zbylých proměnných. K ověření je třeba si uvědomit, že žádný řádek není celý pokryt proměnnými z y_A. Vezmeme-li dvě různá uvažovaná dosazení, která se liší například na řádku i, dokážeme různost získaných subfunkcí tím, že proměnné mimo y_A na řádku i fixujeme na 1 a mimo řádek i na 0. □

Definice 7.33 Nechť $\mathrm{sum}(x) = \sum_{i=0}^{n-1} x_i$. Pak $\mathrm{HWB}(x) = x_{\mathrm{sum}(x)}$, kde x_n formálně dodefinujeme jako $x_n = 0$.

Věta 7.34 *Pro libovolné n platí* $\mathrm{OBDD}(\mathrm{HWB}_n) \geq 2^{n/5-1}$.

Důkaz. Zvolme nějaké uspořádání π proměnných X a předpokládejme, že n je dělitelné 10. Pro libovolné k nechť $A_k = \{\pi(0), \pi(1), \ldots, \pi(k-1)\}$ a $B_k = X \setminus A_k$. Pro libovolná k_0, k_1 nechť $\mathrm{Win}(k_0, k_1) = \{k_1, k_1+1, \ldots, n-k_0\}$.

Lemma 7.35 *Jestliže $k = k_0 + k_1$, pak platí*

$$S(\mathrm{HWB}_n, A_k, B_k) \geq \sum_{i=s(k_0,k_1)-k_0}^{k_1} \binom{s(k_0,k_1)}{i} = \sum_{i=s(k_0,k_1)-k_1}^{k_0} \binom{s(k_0,k_1)}{i},$$

kde $s(k_0, k_1) = |A_k \cap \mathrm{Win}(k_0, k_1)|$.

K důkazu lemmatu uvažme všechna možná rozmístění k_1 jedniček a k_0 nul v množině proměnných A_k. Lze snadno ověřit, že pokud se dvě takováto rozmístění liší na množině $A_k \cap \mathrm{Win}(k_0, k_1)$, pak dávají různé subfunkce na množině B_k. Pro libovolné i, které splňuje $0 \leq i \leq s(k_0, k_1)$, $i \leq k_1$ a $s(k_0, k_1) - i \leq k_0$, existuje

rozmístění, které má na množině $A_k \cap \mathrm{Win}(k_0, k_1)$ právě i jedniček a $s(k_0, k_1) - i$ nul. Zvolme jedno z nich a uvažme všechna rozmístění, která vzniknou permutací fixovaných hodnot na množině $A_k \cap \mathrm{Win}(k_0, k_1)$. Tím dostaneme $\binom{s(k_0,k_1)}{i}$ rozmístění, která dávají různé subfunkce. Součtem těchto počtů pro všechna přípustná i získáme požadovaný odhad.

K dokončení důkazu věty zvolíme $k = \frac{6}{10}n$ a uvážíme následující dvě kombinace k_0, k_1

$$\begin{array}{lll} k_0 & k_1 & \mathrm{Win}(k_0, k_1) \\ \frac{1}{10}n & \frac{5}{10}n & \{\frac{5}{10}n, \ldots, \frac{9}{10}n\} \\ \frac{5}{10}n & \frac{1}{10}n & \{\frac{1}{10}n, \ldots, \frac{5}{10}n\} \end{array}$$

Snadno se ověří, že množina A_k zasahuje do sjednocení obou uvažovaných $\mathrm{Win}(k_0, k_1)$ alespoň $\frac{4}{10}n$ prvky a tedy do některého z nich alespoň $\frac{2}{10}n$ prvky. Alespoň pro jednu z uvažovaných kombinací k_0, k_1 tedy máme $s(k_0, k_1) \geq \frac{2}{10}n$. Navíc, $s(k_0, k_1) \leq |\mathrm{Win}(k_0, k_1)| \leq \frac{4}{10}n$. Z lemmatu pak vyplývá

$$S(\mathrm{HWB}_n, A_k, B_k) \geq \sum_{i=0}^{n/10} \binom{n/5}{i} \geq 2^{n/5-1}.$$

Tím je důkaz dokončen. □

Poznamenejme, že existuje uspořádání π, pro které je π-OBDD$(\mathrm{HWB}_n) \leq 2^{0.2029\,n}$.

Věta 7.36 *Pro* HWB_n *existuje read-once r.d. velikosti nejvýše* $O(n^2)$.

Důkaz. Požadovaný diagram popíšeme jako sekvenční výpočet, který čte každou vstupní proměnnou nejvýše jednou. Počet různých stavů, kterými může tento výpočet projít, je pak horním odhadem velikosti odpovídajícího read-once r.d.

V každém okamžiku výpočtu označme jako k_0 resp. k_1 počet přečtených proměnných s hodnotou 0 resp. 1. Dále, výpočet si pamatuje hodnoty těch dosud přečtených proměnných, jejichž index je v množině $\mathrm{Win}(k_0, k_1)$. Strategie volby testované proměnné je taková, aby v každém okamžiku bylo nutné si pamatovat hodnotu nejvýše jedné dosud čtené proměnné a navíc, aby tato proměnná byla na některém kraji množiny $\mathrm{Win}(k_0, k_1)$. Pokud není pamatována žádná proměnná, vybereme k přečtení proměnnou s nejnižším (nebo nejvyšším) indexem. Pokud je nějaká proměnná pamatována, vybereme k přečtení proměnnou, která je na opačném konci množiny $\mathrm{Win}(k_0, k_1)$. Lze snadno ověřit, že tato volba zaručí, že i v dalším kroku je splněn výše uvedený požadavek. Počet různých stavů uvedeného algoritmu je $O(n^2)$. □