NEUNIFORMNÉ VÝPOČTOVÉ MODELY

BRANCHING PROGRAMY
A ROZHODOVACIE STROMY

BAKALÁRSKA PRÁCA

Peter Koscelanský

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky
Katedra informatiky

9.2.1 Informatika

Vedúci práce: RNDr. Andrej Bebják

Bratislava, 2009

.

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím uvedených zdrojov.

.....

Ďakujem môjmu vedúcemu RNDr. Andrejovi Bebjákovi za pomoc pri hľadaní zaujímavej témy, za odborné vedenie a cenné pripomienky.

Abstrakt

V práci sa zaoberáme neuniformnými výpočtovými modelmi, s dôrazom na branching programy, využívané hlavne na reprezentáciu postupností booleovských funkcií, skúmanie ich vlastností a výpočtovej zložitosti. Analyzujeme najvýznamnejšie výsledky z oblasti zložitosti branching programov, študujeme vplyv rôznych ohraničení na základný model. Pre explicitne definované postupnosti funkcií sa získavanie netriviálnych dolných odhadov považuje za ťažký problém. Prezentované náročné techniky na získavanie dolných odhadov sú aplikované na konkrétne postupnosti funkcií a jazykov. Na príkladoch ilustrujeme limity použiteľnosti a hranice kvality dolných odhadov dokázateľných jednotlivými technikami. Zjednocujúci pohľad na doteraz známe techniky a koncept bariér, ktoré treba prekonať, naznačuje smerovanie v študovanej oblasti.

Kľúčové slová: branching programy, neuniformné výpočtové modely, dolné odhady zložitosti, natural proofs

Obsah

Úvod

Zložitosť booleovských funkcií je v oblasti teoretickej informatiky dobre známy pojem. Prudký rozvoj výpočtovej techniky, ktorej základom je práve efektívna implementácia booleovských funkcií, pôsobil záujem o štúdium ich vlastnosti. Navrhli sa výpočtové modely na ich reprezentáciu. Prirodzene sa začalo pracovať s modelom booleovských obvodov (kapitola 2.2), takmer presne kopírujúcich návrhy skutočných hardvérových obvodov. V teórií ich zložitosti nebol zaznamenaný veľký úspech. Dolné odhady zložitosti explicitne definovaných funkcií sa dokazujú ťažko. Už nelineárne ($\omega(n)$) odhady sa ukázali ako veľmi ťažké a na všeobecných booleovských obvodoch doteraz dôkaz takejto hranice ostáva otvoreným problémom.

Časová zložitosť na Turingových strojoch úzko súvisí so zložitosťou booleovských obvodov (1). Snahou bolo nájsť vhodný model pre reprezentáciu booleovských funkcií, ktorý by zodpovedal priestorovej zložitosti.

Model branching programov (BP), vznikol ako alternatíva k booleovským obvodom na popisovanie booleovských funkcií. Zložitosti BP veľmi dobre vystihujú ohraničenia neuniformných výpočtových modelov na priestorovú zložitosť. Pre BP sa tiež skúmali možnosti ako ukázať dolné odhady. Tu boli snahy o dosť úspešnejšie a podarilo sa dosiahnuť asymptoticky lepšie ako lineárne hranice na BP bez ohraničení. Z teoretického hľadiska majú dôkazy väčších (povedzme exponenciálnych) odhadov pre zložitosti BP veľký význam. V niektorých zložitostných triedach, by takéto odhady zložitosti znamenali rozhodnutie závažných problémov (ako NC^1 vs. L).

Barrington ukázal previazanosť medzi booleovskými obvodmi a BP. Publikoval výsledok (2), podľa ktorého efektívne paralelne riešiteľné problémy na booleovských obvodoch (trieda NC^1) a BP s polynomiálnou veľkosťou a konštantnou šírkou, definujú tú istú triedu (výsledku sa venujeme v kapitole 3.1).

BP neostali iba teoretickým modelom, ujali sa aj v praxi. Ukázalo sa, že tvoria vhodnú dátovú štruktúru na reprezentáciu booleovských funkcií. Efektívne sa na nej dajú robiť operácie ako substitúcia konštanty, konjunkcie a disjunkcie funkcií. Počet rôznych funkcií s danou dĺžkou vstupu je exponenciálny a preto väčšina funkcií musí mať BP s exponenciálnou zložitosťou. V praxi nás zaujímajú hlavne reprezentácie s polynomiálnou veľkosťou a najlepšie s malým stupňom polynómu. Preto je dôležité skúmať BP s ohraničenou veľkosťou. Pomocou dolných odhadov zisťujeme, ktoré triedy funkcií sa nedajú reprezentovať BP s nižšou zložitosťou.

V prvej kapitole práce formálne definujeme branching programy a niektoré základné pojmy. Zaoberáme sa mierami zložitosti na BP. V závere prezentujeme príklady BP pre konkrétne postupností funkcií.

Druhá kapitola sa venuje pojmu neuniformný výpočtový model, keďže BP sú práve takýmto modelom. Ukážeme, prečo neuniformnosť dáva modelu väčšiu výpočtovú silu ako majú Turingové stroje (TS). Ďalej zavádzame pojem neuniformný TS ako analógiu k neuniformným modelom vo svete TS a k nemu definujeme neuniformné rozšírenia klasických zložitostných tried. Uvádzame argumenty, prečo veľkosť BP úzko súvisí práve s priestorovou zložitosťou na TS.

Samozrejme aj na BP sa skúma správanie modelu po pridaní ohraničení. Na zoznámenie s rôznymi snahami v tejto oblasti slúži tretia kapitola. Spomenieme najznámejšie dosiahnuté výsledky, ktoré hovoria o hierarchiách zložitosti takýchto BP. Prezentujeme ekvivalencie medzi triedami definovanými v teórii zložitosti a triedami reprezentovanými BP s ohraničeniami.

Jadro prace tvorí štvrtá kapitola, v ktorej sa venujeme technikám dolných odhadov pre zložitosti explicitne definovaných funkcií na modeli BP. V práci sa snažíme zosumarizovať rôzne techniky. Techník, ktoré priniesli netriviálne odhady na všeobecnom modeli BP, je veľmi málo. Spomíname preto aj techniky na BP s ohraničeniami. Na nich sa vyvinulo množstvo odhadov pre zložitosť booleovských funkcií. Všetky sa držia v podstate niekoľkých základných pozorovaní a práve tieto myšlienky kapitola prezentuje. Pre lepšiu orientáciu sú v chronologickom usporiadaní. Taktiež prináša aj konkrétne príklady použitia jednotlivých techník. V závere podkapitol uvádzame referencie na ďalšie zdroje týkajúce sa danej techniky.

Fakt, že všetky techniky dolných odhadov sa držia rovnakej schémy, si všimli Razborov a Rudich (3). V poslednej kapitole spomíname ich koncept natural proofs, ktorý je do značnej miery zodpovedný za minimálny úspech pri dokazovaní dolných odhadov zložitosti pre explicitne definované booleovské funkcie.

Práca predpokladá, že čitateľ pozná základné pojmy z teórie zložitosti.

Kapitola 1

Základné pojmy

Práca je venovaná najmä branching programom (BP), formálne definujeme výpočtový model. Skúmame tiež niektoré miery zložitosti, ktoré sa bežne definujú na BP. V závere uvedieme príklady ako postupností BP reprezentujú konkrétne formálne jazyky a booleovské funkcie.

Model podobný BP po prvý krát spomenul v roku 1959 Lee vo svojom článku Representation of Switching Circuits by Binary-Decision Programs (4). Nazval ho binárne rozhodovacie programy a použil na reprezentáciu booleovských funkcií vo forme dátovej štruktúry – strom. Model BP, v podobe ako ho prezentujeme v práci, bol zavedený až Masekom (5). Pomenoval ich rozhodovacie grafy (decision graphs). Ich model bol založený na idei konečnostavového stroja s priamym prístupom k vstupu, ktorý v stave môže čítať iba jeden znak zo vstupu. Na základe stavu a prečítaného vstupu sa rozhodne ako bude výpočet pokračovať ďalej.

1.1 Branching programy

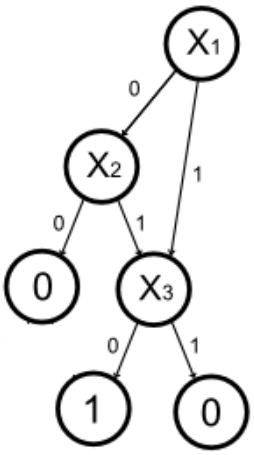
Pre BP existuje viacero definícií, ktoré sa podstatne líšia. Zadefinujeme dve z nich. Prvú uvedieme definíciu BP, ktorú používa väčšina súčasných autorov. Druhá (6) ponúkajúcu praktickejší pohľad na priebeh výpočtu programov.

Definícia 1.1.1: Symbolom B_n označíme triedu všetkých booleovských funkcií $f(x_1, \dots, x_n)$ na n premenných: $\{0,1\}^n \to \{0,1\}$.

Definícia 1.1.2: (7) Branching program (BP) na množine premenných $X_n = \{x_1, \dots, x_n\}$ je orientovaný acyklický graf G s označenými vrcholmi a hranami spĺňajúci nasledujúce vlastnosti

1) G má práve jeden koreň (angl. source)
2) Každý vrchol z G má výstupný stupeň 0 alebo 2
3) Každý vrchol $v \in V(G)$ s výstupným stupňom 2 je označený niektorou premennou x_i, jedna jeho výstupná hrana má hodnotu 1 a druhá hodnotu 0
4) List (angl. sink) je vrchol grafu s výstupným stupňom 0, každý list je označený konštantou 0 alebo 1.

Príklad BP pre konkrétnu booleovskú funkciu je uvedený na Obr. 1.



Obr. 1: Branching program pre funkciu $f(x_1, x_2, x_3) = \overline{x_3}(x_1 \vee \overline{x_1}x_2)$

Ľubovoľný vrchol BP G so vstupnými premennými $x_1, \dots, x_n$ reprezentuje booleovskú funkciu f_v n premenných. Celý BP G s koreňom s reprezentuje funkciu f_s. f_v môže byť definovaná nasledovnými spôsobmi. Konkrétny vstup a je n-tica hodnôt premenných $(a_1, \dots, a_n)$.

Definícia 1.1.3: Výpočet f_v na vstupe $(a_1, \dots, a_n)$ začína vo vrchole v. Ak sa v priebehu výpočtu dostaneme do vrcholu s označením x_i, ďalej pokračujeme po výstupnej hrane s hodnotou a_i. Výsledok $f_v(a)$ zodpovedá hodnote listu, ktorý sme dosiahli.

Definícia 1.1.4: List u s hodnotou b realizuje konštantnú funkciu $f_s(a) \equiv b$. Nech w a z sú priamymi nasledovníkmi vrcholu v, dosiahnuteľní po hrane s hodnotou 0, resp. 1. Nech g a h sú booleovské funkcie reprezentované vrcholmi w (resp. z). Ak v je označený premennou x_i, hodnota funkcie $f_v(a)$ je definovaná podľa Shannonovho pravidla dekompozície[1] ako $f_v(a) \equiv a_i \cdot h(a) \vee \overline{a_i} \cdot g(a)$.

Uvedené definície výpočtu sú navzájom ekvivalentné. Prvá je prístup zhora nadol, kde ideme od koreňa stromu až do niektorého listu. Presná cesta je daná postupnosťou vrcholov, v ktorých sme sa rozhodovali na základe bitu vstupu, či pôjdeme ďalej po hrane 0 alebo 1. Druhá je prístup zdola nahor, kde postupne definujeme listy ako konštantné funkcie 0 a 1. Ostatné vrcholy v vieme definovať pomocou vrcholov v podstrome, ktorého koreňom je v.

V úvode tejto kapitoly sme spomenuli, že BP sa zo začiatku volali binárne rozhodovacie stromy. Odvtedy je označovanie nejednotné. V niektorej literatúre sa pod pojmami branching programy a binárne rozhodovacie stromy rozumie jeden a ten istý model. Inde sa ako rozhodovací strom myslí BP, ktorého graf je strom. Ďalej v práci rozlišujeme medzi týmito pojmami.

[1] Každá booleovská funkcia $f(x_1, \dots, x_n)$ sa dá pre ľubovoľnú premennú x_i rozložiť do tvaru: $f(x_1, \dots, x_i, \dots, x_n) \Leftrightarrow x_i \cdot f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n) \vee \overline{x_i} \cdot f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$, kde súčin je konjunkcia

Definícia 1.1.5: (6) Postupnosť inštrukcií $I_0, I_1, \dots, I_m$ typu $I_j = ite(C_j, T_j, E_j)$ nazveme *ite*[2] program. Kde C_j je podmienka, T_j je *then* vetva a E_j je *else* vetva. C_j, T_j, E_j sú prvkami z množiny $\{0,1, x_1, x_2, \dots, x_n, I_0, I_1, \dots, I_{j-1}\}$ ($x_1, x_2, \dots, x_n$ sú vstupy). Každá inštrukcia $I_j = (x, y, z)$ zodpovedá booleovskej funkcií $f_j \in B_n$ definovanej nasledovne $f_j = xy \vee \bar{x}z$. Branching program je *ite* program, kde podmienky C_j sú premenné z množiny $\{x_1, x_2, \dots, x_n\}$ a vetvy T_j, E_j sú z $\{0,1, I_0, I_1, \dots, I_{j-1}\}$.

Uvedená definícia má už bližšie k programovacím jazykom, lebo *ite* program sa dá ľahko prepísať na postupnosť príkazov GOTO. Je ekvivalentná s definíciou 1.1.2. Každý BP P, ktorého vrcholy sú $v_1, \dots, v_n$ sa dá prepísať na prislúchajúcu postupnosť *ite* inštrukcií. Vnútorný vrchol grafu v_i ohodnotený x_j s nasledovníkmi v_k (po hrane označenej 1) a v_l (po hrane s 0) prepíšeme na inštrukciu $I_i = ite(x_j, I_k, I_l)$. Listy nahradíme za ich ohodnotenia (tomu zodpovedajú inštrukcie $ite(x, 1,1)$ a $ite(x, 0,0)$ pre ľubovoľnú premennú x). Ešte musíme zabezpečiť správne očíslovanie vrcholov, aby sme splnili, že inštrukcia s vyšším číslom odkazuje len na inštrukcie s nižším. To ale nie je problém, lebo graf je orientovaný acyklický. Môžeme jeho vrcholy prečíslovať tak, aby po transformovaní na inštrukcie bola táto podmienka splnená. Listom priradíme najnižšie čísla. Potom postupujeme nasledovne: Ak vrchol má oboch nasledovníkov už očíslovaných, priradí sa mu najnižšie ešte nepoužité číslo.

Opačne, každú inštrukciu vieme simulovať jedným vrcholom v grafe. Konštrukcia je rovnaká ako vyššie, akurát inštrukcie transformujeme na vrcholy.

Ite program korešpondujúci s BP na Obr. 1 pozostáva zo šiestich inštrukcií $I_0, \dots, I_5$. Listy BP transformujeme na inštrukcie $I_0 = (x_1, 0,0)$, $I_1 = (x_1, 1,1)$ a $I_2 = (x_1, 0,0)$. K vnútorným vrcholom prislúchajú $I_3 = (x_3, I_2, I_1)$, $I_4 = (x_2, I_3, I_0)$ a $I_5 = (x_1, I_3, I_4)$.

Definujeme reprezentáciu formálneho jazyka na BP. Obmedzíme sa len na jazyky nad dvojznakovou abecedou (0, 1). Môžeme prirodzene rozšíriť definície BP na ľubovoľnú abecedu. BP bude mať z každého vrcholu viac možností ako pokračovať ďalej vo výpočte.

Definícia 1.1.6: Charakteristickou funkciou množiny $A \subseteq \{0,1\}^n$ nazveme booleovskú funkciu $f \in B_n$ takú, že $f(x_1, \dots, x_n) = 1$ práve vtedy, keď reťazec $x_1 \dots x_n \in A$.

Definícia 1.1.7: Postupnosť branching programov $\{P_i\}_{i=0}^{\infty}$ definuje formálny jazyk $L \subseteq \{0,1\}^*$, ak funkcia $f_i \in B_i$ reprezentovaná P_i je charakteristickou funkciou množiny $L \cap \{0,1\}^i$.

[2] *if-then-else*

1.2 Miery zložitosti

Na BP nás, podobne ako na každom inom výpočtovom modeli, zaujímajú miery zložitosti. Prirodzenými mierami na Turingových strojoch sú časová ($TIME$) a priestorová ($SPACE$) zložitosť. Na BP budeme definovať ich ekvivalenty – hĺbku (angl. depth) a veľkosť (angl. size).

Definícia 1.2.1: Veľkosť BP P je rovná počtu vnútorných vrcholov v P. Hĺbka je maximum zo všetkých dĺžok ciest od koreňa P k listom.

Veľkosť branching programu P označujeme $BP(P)$ a hĺbku $BPD(P)$. Podobne pre booleovskú funkciu f, budeme symbolmi $BP(f)$ a $BPD(f)$ označovať minimálnu veľkosť (resp. hĺbku) BP, ktorý realizuje funkciu f.

Ak $\{f_n\}_{n=0}^{\infty}$ je postupnosť funkcií, potom veľkosť BP pre booleovskú funkciu $f_n \in B_n$ je závislá od parametra n. Preto veľkosť jednotlivých BP pre booleovské funkcie z postupnosti je funkcia s premennou n.

Symbolom PBP budeme označovať triedu všetkých funkcií, pre ktoré existuje BP s polynomiálnou veľkosťou od dĺžky vstupu.

Ak si BP predstavíme ako procesor, ktorý sa rozhoduje na základe jedného bitu vstupu (definícia 1.1.5), potom hĺbka odzrkadľuje čas (počet operácií), ktoré by musel vykonať takýto procesor aby dospel k výstupu. Veľkosť hovorí o tom, koľko inštrukcií musí mať v pamäti procesor aby mohol realizovať danú funkciu, preto sa dá chápať ako akýsi ekvivalent priestoru.

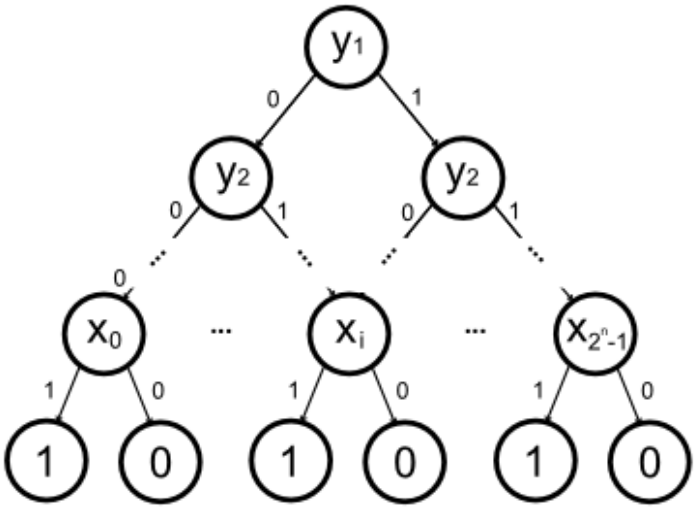
Menej častou mierou je šírka. Pri jej definovaní sa zvykne používať BP s vrstvami (leveled-BP).

Definícia 1.2.2: Leveled-BP P je BP, kde sa všetky vrcholy dajú usporiadať do vrstiev tak, že hrany z jednej vrstvy smerujú iba do nasledujúcej vrstvy.

Definícia 1.2.3: Šírka leveled-BP P je maximum z počtu vrcholov na jednotlivých vrstvách. Triedu BP (aj funkcií nimi reprezentovanými) so šírkou n označíme ako W_n.

1.3 Branching programy pre konkrétne postupnosti funkcií

Príklad 1: Funkcia storage access $SA_n(x, y)$ je definovaná na $2^n + n$ premenných ($|x| = 2^n$, $|y| = n$). Výsledkom $SA_n(x, y)$ je $x_{bin(y)}$ [3], teda vlastne bit z x, ktorý je na pozícií označenej hodnotou čísla y (číslovanie bitov x začína od 0). BP P pre takýto jazyk z týchto funkcií najprv zistí číslo zo vstupu y a následne vráti požadovanú hodnotu. Bude mať tvar úplného binárneho stromu o výške n. V liste zodpovedajúcom y bude mať príslušnú premennú $x_{bin(y)}$. Odtiaľ pôjde hrana 1 do listu s označením 1 a hrana 0 do listu s označením 0. Zjavne takto popísaný BP realizuje funkciu SA_n. Štruktúra programu je znázornená na Obr. 2.

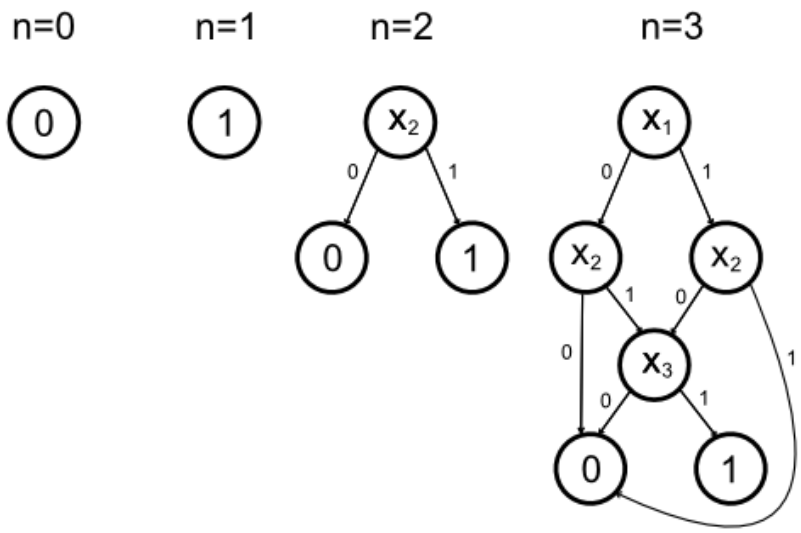


Obr. 2: Náčrt branching programu pre funkciu SA_n.

Pre hĺbku P platí $BPD(P) = O(n)$, n ale nie je dĺžka vstupu, ale iba parameter funkcie. Preto ak ako m označíme počet vstupov pre SA_n, dostaneme $m = 2^n + n$ a teda $BPD(P) = O(\log m)$. Podobne pre veľkosť $BP(P) = O(2^n) = O(m)$. Program P je vlastne leveled, lebo na i-tej ($i \leq n$) úrovni v strome sa testuje bit y_i a na poslednej sa testujú bity z x. Preto má zmysel hovoriť o šírke programu, ktorá má hodnotu $O(2^n) = O(m)$.

Príklad 2: Jazyk $PRIME_n$ je definovaný na vstupoch x dĺžky n a platí $PRIME_n(x) = 1$ práve vtedy ak číslo $bin(1x)$ (číslo v binárnom zápise začínajúce cifrou 1 a pokračujúce ciframi x) je prvočíslo.

[3] $bin(y)$ je číslo reprezentované binárnym reťazcom y



Obr. 3: Začiatok postupností BP pre jazyk prvočísel $PRIME_n$.

V predchádzajúcom príklade sa dali jednoducho zostrojiť BP pre rôzne dĺžky vstupu, v tomto to už nie je také jednoduché. Ako uvádzame na začiatku nasledujúcej kapitoly (lema 2.1.2), každý jazyk sa dá akceptovať postupnosťou BP, ktorá má veľkú zložitosť. Všeobecnou konštrukciou by sme vedeli zostrojiť BP, tie ale s veľkou pravdepodobnosťou nebudú mať optimálnu veľkosť pre náš jazyk $PRIME_n$, keďže sú to stromy. Začiatok postupnosti BP pre tento jazyk je na Obr. 3. Pre $n = 3$ už program nie je leveled a nie je to ani strom. Ak by sme trvali na prísne stromovej štruktúre, BP by mal určite viac vrcholov.

Príklad 3: Jazyk L definujeme ako $L = \{w \in \{0,1\}^* | \#_1(w) \geq |w|/2\}$. L je jazyk všetkých reťazcov, v ktorých je počet jednotiek väčší nanajvýš rovný počtu núl. Ukážeme, že existuje postupnosť BP $\{P_i\}_{i=0}^{\infty}$, ktorá definuje jazyk L. Podľa definície 1.1.7 musíme zostrojiť jednotlivé P_n tak, že funkcia f_n reprezentovaná P_n je charakteristickou funkciou množiny $L \cap \{0,1\}^n$. f_n je funkcia, ktorá na vstupe a vracia hodnotu 1 práve vtedy, keď a obsahuje aspoň $\lceil n/2 \rceil$ bitov nastavených na 1. P_n pre f_n bude mať $n + 1$ vrstiev, na každej vrstve bude najviac $n + 1$ vrcholov. Tieto vrcholy (v rámci jednej vrstvy) budú zodpovedať počtu 1 v bitoch prečítaných pred dosiahnutím danej vrstvy. Vo všetkých vrcholoch na prvej vrstve sa testuje premenná x_1, na druhej x_2, podobne pre ďalšie vrstvy. Posledná vrstva sú listy, každý list má priradené číslo z $(0, \dots, n)$ určujúce počet 1 na vstupe. Listy s priradeným číslom väčším nanajvýš rovným $\lceil n/2 \rceil$ budú označené konštantou 1, ostatné konštantou 0. Ľahko vidieť, že takýto BP realizuje funkciu f_n. Postupnosť BP $\{P_i\}_{i=0}^{\infty}$ teda definuje formálny jazyk L.

Kapitola 2

Neuniformné výpočtové modely

Jedna zo základných vlastností BP je neuniformnosť. Faktom je, že v tom spočíva podstata odlišnosti BP od klasických výpočtových strojov (napr. Turingove stroje), preto venujeme neuniformným modelom celú kapitolu. Pri modeli TS je na definovanie konkrétneho stroja najdôležitejšie zadať δ-funkciu. Tá popisuje ako sa má správať stroj. Ak je v stroj nejakom stave q a z pásky číta symbol a, potom prechodová funkcia $\delta(q, a)$ určuje ako sa zmení stav, čím sa má na páske nahradí symbol a a kam sa posinie čítacia hlava. δ-funkcia je nezávislá na konkrétnom vstupe. Používa sa pre všetky dĺžky vstupov. Teda je uniformná.

Naproti tomu, pri neuniformných modeloch nie je jedno pre akú dĺžku vstupu sa realizuje výpočet. Z definície BP (kapitola 1.1) je zrejmé, že konkrétny BP je navrhnutý len pre vstupy rovnakej dĺžky. BP odkazuje na premenné priamo, čoho dôsledkom je obmedzenie na fixnú dĺžku vstupu. Preto sme v definícií 1.1.7 na reprezentáciu jazyka použili postupnosť BP, pre každú dĺžku vstupu jeden program. Tým sme zaručili možnosť realizovať výpočet pre ľubovoľný vstup a dospieť k jeho akceptácií (výsledkom je 1) alebo k zamietnutiu (výsledkom je 0). Stačí nájsť v postupnosti zodpovedajúci BP k vstupu zadanej dĺžky. Prináša to so sebou aj inú podstatnú vlastnosť. BP v postupnosti sa nemusia na seba vôbec podobať, môžu byť úplne odlišné, čiže neuniformné. To je značný rozdiel oproti uniformným modelom. Tie musia na každom vstupe rátať s rovnakým predpisom (pri TS δ-funkcia). Neuniformné modely si môžu pre rôzne dĺžky vstupov dovoliť rátať úplne rozdielne.

V nasledujúcej časti ukážeme, že toto naozaj pomáha a sila neuniformných modelov (napríklad BP) je väčšia ako výpočtová sila TS.

2.1 Popisná sila branching programov

Vieme, že TS bez ohraničení akceptujú triedu jazykov, ktorá sa označuje $\mathcal{L}_{RE}$ - rekurzívne vyčísliteľné jazyky. Obsahuje aj veľmi netriviálne problémy ako napríklad problém zastavenia. Jednoduchým argumentom sa dá ukázať existencia jazyka nepatriaceho do $\mathcal{L}_{RE}$. Stačí nám nato obyčajné spočítanie veľkosti množín. Všetkých jazykov je nespočítateľne veľa, TS je iba spočítateľne veľa (vieme ich zoradiť lexikograficky do postupnosti). Preto musí existovať jazyk, ktorý sa nedá rozpoznávať na žiadnom TS. Ukážeme, že pomocou BP sa dá reprezentovať ľubovoľný jazyk.

Lema 2.1.1: Pre každú funkciu $f \in B_n$ (množina všetkých booleovských funkcií nad n premennými), existuje BP, ktorý túto funkciu realizuje.

Dôkaz: Indukciou vzhľadom na n (počet premenných). Báza indukcie je z definície 1.1.4 jasná. BP vedia realizovať ľubovoľnú booleovskú funkciu z B_0 – konštantnú funkciu. To budú iba samotné listy. Predpokladajme teda, že vieme urobiť BP pre všetky funkcie z množín $B_0, B_1, \dots, B_{n-1}$. Funkciu $f(x_1, \dots, x_n) \in B_n$ dekomponujeme pomocou Shannonovho pravidla na tvar $x_1 \cdot f(1, x_2, \dots, x_n) + \overline{x_1} \cdot f(0, x_2, \dots, x_n)$. Funkcie $f(1, x_2, \dots, x_n) = f_1$ a $f(0, x_2, \dots, x_n) = f_0$ sú vlastne booleovské funkcie nad $n-1$ premennými, preto máme podľa indukčného predpokladu pre ne BP. Podľa definície ak máme BP pre funkcie f_0 a f_1 potom vieme urobiť aj BP pre ich kompozíciu v premennej x_1, čo je funkcia f. ■

Pre definovanie jazyka sa používa postupnosť BP, z ktorých každý realizuje charakteristickú funkciu množiny slov jazyka s konkrétnou dĺžkou. Čiže pomocou lemy vyššie ľahko ukážeme, že BP vedia definovať ľubovoľný (aj nie rekurzívne vyčísliteľný) jazyk.

Lema 2.1.2: Nech L je ľubovoľný jazyk nad binárnou abecedou, potom k nemu existuje postupnosť branching programov $\{P_i\}_{i=0}^{\infty}$, ktorá ho reprezentuje.

Dôkaz: Pre každé j definujeme booleovskú funkciu f_j na j premenných ako charakteristickú funkciu $L \cap \{0,1\}^j$. Podľa lemy 2.1.1 sa dá k tejto funkcií zostrojiť BP P_j. Zjavne postupnosť takto definovaných BP reprezentuje jazyk L. ■

Vidíme že neuniformné modely majú výpočtovú silu ešte väčšiu ako uniformné. BP ale nie sú jediným skúmaným neuniformným modelom. Ešte pred BP sa zaviedli booleovské (logické) obvody. Keďže sa na ne v práci niekoľkokrát odvolávame, v nasledujúcej časti ich zadefinujeme.

2.2 Booleovské obvody

Model booleovských obvodov (boolean circuit) tvorí vo svete výpočtových modelov analógiu k obvodom v hardvéri. Pozostáva z logických hradiel, čo sú vlastne booleovské funkcie s jedným výstupom. Hradlá sú navzájom rôzne poprepájané. Bity vstupu a sú privedené ako vstup na niektoré hradlá. Výstupom sú výstupy z hradiel, ktoré už nevedú do žiadneho hradla. Evidentne je dôležité ako zvolíme funkcie, ktoré môžu byť hradlami. Tieto funkcie sa nazývajú báza a najčastejšie sa používajú také, ktoré tvoria úplný systém. Ako napríklad $\{\wedge, \vee, \neg\}$.

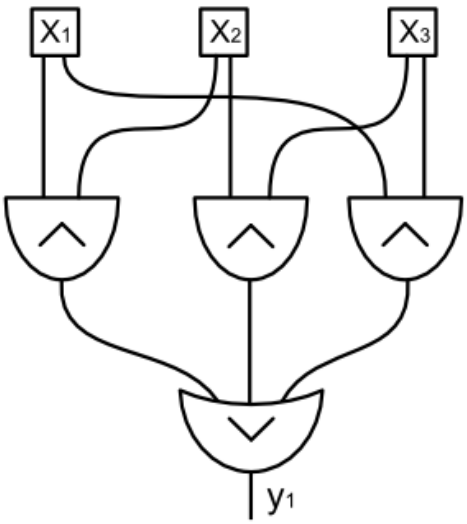
Definícia 2.2.1: (1) Nech Ω je báza funkcií s jedným výstupom. Booleovský obvod nad bázou Ω funguje pre konkrétne n na vstupe $x_1, \dots, x_n$. Obvod je zložený z konečného počtu hradiel $G(1), \dots, G(t)$. Hradlo $G(i)$ pozostáva z typu $\omega_i \in \Omega$ a predchodcov. Ak funkcia $\omega_i \in B_{N(i)}$, potom predchodcovia sú $N(i)$-tica $\big(P(1), \dots, P\big(N(i)\big)\big)$. $P(i)$ je

prvkom z množiny $\{0,1,x_1,\dots,x_n,G(1),\dots,G(i-1)\}$. Pomocou $res_{G(i)}(x)$ označíme funkciu realizovanú v $G(i)$ na vstupe x, potom výstup obvodu definujeme induktívne. Ak I je vstup, potom res_I je rovné I. Ak $G(i)=\left(\omega_i,\left(P(1),\dots,P\left(N(i)\right)\right)\right)$, potom $res_{G(i)}(x)=\omega_i\left(res_{P(1)},\dots,res_{P\left(N(i)\right)}\right)$. Výstupný vektor je $(y_1,\dots,y_m)$, kde y_i je buď vstup, alebo hradlo. Funkcia, ktorú celý obvod realizuje je f s n vstupmi a m výstupmi. $f=(f_1,\dots,f_m)$, f_i je funkcia realizovaná v y_i.

Obvody si môžeme, podobne ako BP, predstaviť ako orientovaný acyklický graf. Jednotlivé vrcholy sú funkcie z bázy. Do listov privedieme vstupy a konštanty $(0,\ 1)$, potom postupne vyhodnocujeme funkcie vo vrcholoch, ktoré majú už všetky vstupy známe (takto vyhodnotíme najprv vrcholy „nad“ listami a pokračujeme až k výstupom z obvodu). Reprezentácia formálneho jazyka je definovaná ako pri BP (definície 1.1.6 a 1.1.7), použije sa pri nej model obvodu s jedným výstupným bitom. Je zrejmé, že jeden obvod slúži iba pre konkrétnu dĺžku vstupu, preto sú obvody neuniformný výpočtový model.

Na obvodoch sa definujú miery popisujúce zložitosti. Veľkosťou obvodu C sa označuje počet hradiel v obvode. Hĺbka C označuje najdlhšiu (v počte hradiel) cestu od niektorého vstupného bitu k výstupu. $C(f)$ označuje veľkosť najmenšieho obvodu pre f. Väčšinou sa používajú obvody s jedným výstupným bitom a s ohraničeným vstupným stupňom 2, čo znamená, že hradlá nemajú viac ako 2 vstupy.

Príklad booleovského obvodu C pre funkciu $f(x_1,x_2,x_3)$ je načrtnutý na Obr. 4. Funkcia f je charakteristickou funkciou množiny $L\cap\{0,1\}^3$ pre jazyk L z príkladu 3 v kapitole 1.3. $f(x_1,x_2,x_3)=1$ práve vtedy, keď počet vstupných bitov nastavených na 1 je aspoň 2. Preto k f zodpovedá formula $x_1x_2\vee x_2x_3\vee x_1x_3$. Booleovský obvod C sa skladá z dvoch vrstiev. Vstupy sú po dvoch privedené do hradiel reprezentujúcich



Obr. 4: Booleovský obvod pre vstup dĺžky 3, s veľkosťou 4 a hĺbkou 2

konjunkcie. Výstupy z týchto hradiel sú vstupom pre hradlo reprezentujúce disjunkciu. Ľahko vidieť, že takýto obvod definuje funkciu f. V obvode sú 4 hradlá, hĺbka je rovná 2. Hradlo s disjunkciou má vstupný stupeň 3. Ak trváme na hradlách s dvoma vstupmi, potom musíme toto hradlo nahradiť dvoma disjunkciami. Zvýši sa tým veľkosť aj hĺbka obvodu.

Neuniformnosť nemusí vždy vyhovovať a preto sa zaviedli uniformné obvody. Obvodu ľahko priradíme kód. Každému hradlu prislúcha štvoricu (i, l, r, t), kde i je poradové číslo hradla, l (resp. r) je číslo ľavého (resp. pravého) predchodcu a t je číslo funkcie z bázy (efektívne ich očíslujeme). Postupnosť obvodov bude uniformná ak sa dá takýto kód generovať na TS.

Definícia 2.2.2: (1) Postupnosť booleovských obvodov $S = \{C_n\}_{n\geq 0}$ (c_n označuje veľkosť C_n) je BC-uniformná[4], ak existuje TS, ktorý zo vstupu 1^n vygeneruje kód pre C_n a jeho priestorová zložitosť je ohraničená $O(\log c_n)$.

Podobne sa dajú definovať aj iné typy uniformít, väčšinou zmenou kódu alebo ohraničení na TS. Takéto „zuniformňovanie“ sa dá spraviť aj pri BP.

V rámci obvodov sa definovali triedy NC^i (Nick’s class), ktoré sa stali triedami efektívne paralelne riešiteľných problémov. Dôvod je jednoduchý. Vyhodnotenie obvodu C na počítači sa robí v čase, ktorý zodpovedá hĺbke obvodu. Pritom sa musí ísť paralelne (od každého bitu vstupu súčasne).

Definícia 2.2.3: Trieda NC^i je trieda tých problémov, ktoré sa dajú riešiť na booleovských obvodoch s hĺbkou $O\left(\log^{\mathrm{i}} n\right)$, veľkosťou $n^{O(1)}$ a ohraničeným vstupným stupňom 2. $NC = \bigcup_{i\geq 0} NC^i$

2.3 Neuniformný Turingov stroj

Postupnosť BP (resp. booleovských obvodov) môže pre rôzne dĺžky vstupov použiť rôzne programy. To je dôvod prečo sa nedajú BP simulovať na bežnom TS. Samotný výpočet by nebolo ťažké prebehnúť aj na TS. Takýto TS by musel poznať istý kód BP pre vstup konkrétnej dĺžky, aby mohol odsimulovať výpočet. Problém je práve zostrojiť tento kód. BP v postupnosti na definovanie jazyka môžu byť natoľko rôzne (neuniformné), že sa ich kódy nedajú s konečným počtom stavov generovať. V predošlej časti sme spomenuli možnosť ako sa z obvodov a BP dajú urobiť uniformné modely. Tie sa už dajú ľahko simulovať na TS, keďže stroj môže ich kód generovať. Dá sa ísť na to aj opačne, pridáme k TS niečo, čo z neho spraví neuniformný model. Zaviedol sa model neuniformného TS (8), tiež sa označuje ako TS s nápoveďou. K stroju sa pridáva k stroju referenčnú pásku (orákulum), na ktorej môže mať čokoľvek, čo radí pri výpočte.

[4] podľa autorov Borodin, Cook

Definícia 2.3.1: Neuniformný Turingov stroj je Turingov stroj[5] s jedným orákulom – páskou iba na čítanie, ktorej obsah závisí len od dĺžky vstupu a nie od vstupu samotného. Priestorová zložitosť takéhoto stroja je súčet miesta použitého na pracovnej páske a logaritmu z dĺžky orákula.

Nezaujíma nás obsah orákula ale len jeho dĺžka. Preto je použitý logaritmus dĺžky orákula pri definícií priestorovej zložitosti.

Cobham vo svojej práci (9), prvý krát ukázal vzťah medzi priestorovou zložitosťou neuniformného TS a veľkosťou branching programu. (Symbolom $S_f(n)$ označíme priestorovú zložitosť neuniformného deterministického TS, ktorý vyhodnocuje postupnosť funkcií $f = \{f_n\}$, kde $f_n \in B_n$)

Veta 2.3.1: (6) Nech $f = \{f_n\}$ je postupnosť booleovských funkcií $f_n \in B_n$, potom platí:

1) $S_f(n) = O(\log s_1(n))$, kde $s_1(n) = \max\{BP(f_n), n\}$
2) $BP(f_n) = 2^{O(s_2(n))}$, kde $s_2(n) = \max\{S_f(n), \log n\}$

Keďže booleovské funkcie $f \in B_n$, ktoré závisia od všetkých premenných[6], majú BP o veľkosti aspoň n a TS so sublogaritmickou pamäťovou zložitosťou vedia riešiť len veľmi jednoduché problémy, môžeme výsledok vety zhrnúť do rovnosti $S_f(n) = \theta(\log BP(f_n))$. V úvode práce sme spomenuli, že BP zachytávajú priestor na TS. Práve o tom hovorí veta. Deterministický priestor na TS je ekvivalentný logaritmu veľkosti BP pre danú postupnosť funkcií.

Dôkaz: Nech P_n je BP o veľkosti $BP(f_n)$ pre funkciu f_n. P_n budeme simulovať pomocou neuniformného TS na vstupe dĺžky n. Ako obsah orákula zvolíme kód P_n. Kód budú tvoriť štvorice, kde bude zapísané číslo vrcholu (nato potrebujeme $O(\log BP(f_n))$ bitov), čísla nasledovníkov v poradí po hranách 0, 1 (obe po $O(\log BP(f_n))$) a číslo premennej zo vstupu ($O(\log n)$), na ktorú sa odkazuje vrchol. Listy budú mať špeciálne kódy, kde bude číslo vrcholu, nasledovníkom dáme pevnú hodnotu 0 a namiesto indexu vstupu tam bude označenie listu (0, 1).

Takýchto štvoríc bude na páske $O\big(BP(f_n)\big)$, preto dĺžka orákula bude $O\big(BP(f_n)(\log BP(f_n) + \log n)\big) = O\big(BP(f_n)(\log\ s_1(n))\big)$. Stroj najprv prekopíruje na pracovnú pásku štvoricu s koreňom (tú si dáme na orákulum ako prvú), potom dekóduje čomu sa rovná bit vstupu, ktorý prislúcha koreňu. Podľa neho vyberie číslo nasledovníka, pohľadá na orákule štvoricu, kde je vrchol s príslušným číslom, a prekopíruje ju na pracovnú pásku. Znova nájde príslušný bit na vstupe a nasledovníka. Takto pokračuje, až narazí na list, vtedy podľa jeho hodnoty akceptuje (výsledkom je 1), alebo zamietne (0).

---

[5] Uvažujeme definíciu TS so vstupom len na čítanie a s oddelenou pracovnou páskou.

[6] Formálne, neexistuje premenná x_i taká, že $f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$ pre ľubovoľné ohodnotenie premenných $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$.

Na pracovnej páske potrebuje TS miesto $O(\log BP(f_n) + \log n) = O(\log\ s_1(n))$. Celková pamäťová zložitosť bude súčet logaritmu použitého miesta orákulom a miesta použitého na pracovnej páske.

$$\log\left(O\left(BP(f_n)(\log\ s_1(n))\right)\right) + O(\log\ s_1(n))$$
$$= O(\log(BP(f_n)\log\ s_1(n))) + O(\log\ s_1(n))$$
$$= O(\log BP(f_n) + \log\log s_1(n)) + O(\log\ s_1(n)) = O(\log\ s_1(n))$$

Opačne, nech M je neuniformný TS pre vstupy dĺžky n vyhodnocuje f_n s priestorom $S_f(n)$. Označme F počet stavov M a Γ počet symbolov pracovnej abecedy stroja M. Orákulum nie je dlhšie ako $2^{S_f(n)}$, lebo logaritmus miesta použitého orákulom je menší ako $S_f(n)$. Počet konfigurácií na pracovnej páske je $\Gamma^{S_f(n)}$, počet možných pozícií hlavy na vstupe je n a na pracovnej páske $S_f(n)$. Celkový počet konfigurácií sa dá odhadnúť ako

$$F \cdot n \cdot S_f(n) \cdot 2^{S_f(n)} \cdot \Gamma^{S_f(n)} = 2^{O(s_2(n))}$$

Každý vrchol BP, ktorý bude simulovať M na vstupoch dĺžky n, bude reprezentovať jednu konfiguráciu M. Koreňom bude počiatočná konfigurácia, akceptačné konfigurácie (M dá výstup 1) budú listy s označeným 1 a zamietacie konfigurácie (M dá 0) listy s 0. Ak v nejakej konfigurácií A číta zo vstupu i-te políčko, potom príslušný vrchol bude mať označenie x_i, nasledovníkom po hrane 0 bude vrchol zodpovedajúci konfigurácií, ktorá nasleduje po A, ak je i-te políčko 0, pre 1 podobne. Nemôže sa nám stať, že takto vytvorený graf bude cyklický. To by znamenalo zacyklenie nášho stroja M, čo je v spore s tým, že M je deterministický a realizuje funkciu f_n. Takto vytvorený BP počíta funkciu f_n na vstupe dĺžky n a má požadovanú veľkosť. ■

Podobne sa dajú simulovať aj booleovské obvody. Na orákulum si vyberieme kód obvodu a simulujeme výpočet.

2.4 Triedy $X/Poly$

Na klasických (uniformných) TS sa zavádzajú rôzne zložitostné triedy pomocou ohraničení. Ohraničenia môžu byť na časovú, alebo priestorovú zložitosť.

Definícia 2.4.1: Trieda P (resp. NP) je trieda tých problémov (jazykov), ktoré sa dajú riešiť na deterministickom (resp. nedeterministickom) TS, ktorý na každom vstupe x dĺžky n dospeje k výsledku (akceptovanie, alebo zamietnutie) v čase (počte krokov) menšom ako $p(n)$. Kde p je nejaký polynóm.

Pomocou týchto tried sa definuje aj najznámejší otvorený problém $P =? NP$. Teda či deterministické a nedeterministické TS v polynomiálnom čase akceptujú tú istú triedu jazykov.

Definícia 2.4.2: L (resp. NL) je trieda jazykov rozpoznávaná deterministickým (resp. nedeterministickým) TS, ktorý na vstupoch x dĺžky n nepoužije viacej ako $O(\log n)$ miesta na páske.

K týmto známym triedam sa dajú definovať neuniformné ekvivalenty. Pomocou pridania orákula.

Definícia 2.4.3: Nech X je trieda jazykov definovaná TS s nejakými ohraničeniami (napr. P). Potom $X/Poly$ je trieda jazykov definovaná neuniformnými TS s tými istými ohraničeniami, ktoré používajú orákulom s dĺžkou ohraničenou $n^{O(1)}$, kde n je dĺžka vstupu.

Čiže $P/Poly$ je trieda jazykov rozoznávaných neuniformnými TS pracujúcimi v polynomiálnom čase, ktorých orákulum nemá väčšiu ako polynomiálnu dĺžku. Na záver uvedieme vetu, ktorá vystihuje BP s polynomiálnou veľkosťou (teda efektívne) v hierarchii neuniformných tried $X/Poly$.

Veta 2.4.1: (9) Triedy PBP a $L/Poly$ sú ekvivalentné.

Dôkaz: Ako dôkaz vety 2.3.1. Keďže máme BP s polynomiálnou veľkosťou, ich kód sa zmestí na orákulom. Uloženie kódu jedného vrcholu na pracovnú pásku potrebuje $O(\log\ n)$ miesta. Preto stroju M, ktorý má tento BP simulovať, stačia obmedzené zdroje z $L/Poly$. Naopak, počet konfigurácií stroja M z triedy $L/Poly$ sa dá ohraničiť polynomiálne, preto stačí na simuláciu BP s polynomiálnou veľkosťou. ■

Viac vzťahov medzi triedami $X/Poly$ a neuniformnými modelmi (ako BP, booleovské obvody, switching networks...) sa dá nájsť v úvode (10).

Kapitola 3

Branching programy s ohraničeniami

V tejto kapitole sa pozrieme nato čo sa stane s popisnou silou BP ak ohraničíme niektorý zdroj. Základné zdroje, ktoré môžeme ohraničiť sú spomenuté v definícií 1.2.1. Je to veľkosť a hĺbka. Lema 2.1.2 nám hovorí, že každý formálny jazyk (funkcia) sa dá na BP realizovať. BP, ktorý sa v dôkaze použije, má hĺbku n a veľkosť 2^n. Čiže nemá zmysel skúmať ohraničenia veľkosti slabšie ako exponenciálne. Z vety 2.4.1 vidíme, ako sa správajú BP s polynomiálnou veľkosťou. Na druhej strane obmedziť veľkosť až pod n už prinesie značné zredukovanie popisnej sily. Dôvodom je fakt, že BP pre funkciu z B_n, ktorá závisí od všetkých premenných, potrebuje aspoň n vrcholov. Ďalej sa preto pozrieme na ohraničenia šírky a počtu čítaní jedného bitu vstupu.

3.1 Branching programy s ohraničenou šírkou

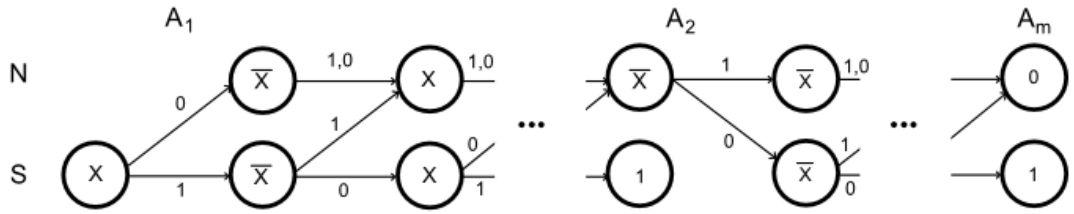
Zjavne BP s ohraničenou šírkou na 1 dokážu popísať iba triviálne funkcie, lebo by v grafe existovala práve jedna cesta, po ktorej musí výpočet postupovať. Ale už aj najmenšie zmysluplné ohraničenie šírky stačí na definovanie ľubovoľného jazyka.

Lema 3.1.1: Do triedy funkcií reprezentovaných pomocou BP so šírkou 2 (W_2) patria všetky booleovské funkcie (formálne jazyky).

Dôkaz: Podobne ako pri leme 2.1.2 nám stačí ukázať, že ku každej funkcií $f \in B_n$ existuje BP z W_2, ktorý túto funkciu realizuje. K funkcií f existuje booleovská formula A v disjunktívnom normálnom tvare, ktorá popisuje túto funkciu.

$$A = A_1 \vee A_2 \vee \ldots \vee A_m$$

Kde A_i je konjunkcia literálov. Nech v A_i je k literálov, potom BP pre A_i bude mať $k + 1$ úrovní. Dva vrcholy na jednej úrovni budú mať za úlohu simulovať, či môže byť klauzula A_i ešte splnená, alebo už nie. Vrcholy prvého typu plus akceptačný vrchol na $k + 1$ úrovni budú tvoriť množinu S a druhého spolu so zamietacím vrcholom množinu N. Na j-tej úrovni sa bude testovať vstup pre j-ty literál z A_i. Z vrcholu v množine N pôjdu obe výstupné hrany do vrcholu z N na $j + 1$ úrovni. Z vrcholu v S pôjdu hrany na $j + 1$ vrstvu podľa toho či je j-ty literál kladný (vtedy pôjde 0 hrana do vrcholu z N a 1 do S), alebo záporný (naopak). Ľahko vidieť, že takýto BP realizuje funkciu s predpisom A_i a jeho výpočet vždy končí na poslednej úrovni.



Obr. 5: Príklad vytvárania BP so šírkou 2 pre formulu v disjunktívnom normálnom tvare. Symbolom X (resp. $\bar{X}$) je označená niektorá premenná, ktorá sa vyskytuje ako kladný (resp. záporný) literál v A_i.

BP P pre f získame zložením týchto čiastočných BP. Akceptačné vrcholy necháme akceptačnými aj v P. Začneme z BP pre A_1, namiesto jeho zamietacieho vrcholu dáme koreň BP pre A_2 a tak ďalej aj po posledný BP pre A_m, kde už necháme aj zamietací vrchol. Výsledný BP je zobrazený na Obr. 5. Zjavne BP P realizuje disjunkciu $A_1 \vee \ldots \vee A_m$, preto P realizuje f. ■

Dlho sa myslelo, že obmedziť šírku na konštantu takmer pre všetky funkcie znamená exponenciálnu zložitosť. Podarilo sa to vyvrátiť Barringtonovi.

Veta 3.1.2: (2) Triedy PBP s konštantnou šírkou ($W_{O(1)}$) a neuniformné NC^1 sa rovnajú.

Barrington na dôkaz tejto vety použil špeciálny model permutačných BP. Ukázal, že permutačné BP so šírkou 5 a polynomiálnou hĺbkou definujú tú istú triedu ako neuniformné NC^1 a PBP s konštantnou šírkou. Veta môže byť prevedená aj na uniformný charakter.

Je známy výsledok $NC^1 \subseteq L$ (11), nevie sa či sú tieto triedy rozdielne. Ich vzťah sa dá konvertovať do reči BP. Veta 2.4.1 hovorí, že PBP a L je tá istá trieda. Ak by sa teda podarilo ukázať, že každý BP z PBP sa dá simulovať BP s ohraničenou šírkou, pri polynomiálnom náraste veľkosti. Potom by sa podľa vety 3.1.2 ukázala rovnosť NC^1 a L. Naopak, ak by sa podarilo ukázať superpolynomiálny dolný odhad na veľkosť BP s konštantnou šírkou pre funkciu reprezentovanú BP z PBP, potom by sa ukázalo $NC^1 \neq L$. Stále je to otvorený problém.

3.2 Read k-times only branching programy

Snaha bola nájsť prirodzenú hierarchiu definovanú pomocou ohraničení na BP. Na booleovských obvodoch dobre funguje hierarchia NC^i, formálne definovaná v kapitole 2.2. Na BP môžeme obmedziť početnosť čítania jedného bitu vstupu počas výpočtu.

Definícia 3.2.1: Branching program P je read k-times only branching program (BP_k), ak každá premenná zo vstupu je na ľubovoľnej ceste od koreňa k listom testovaná najviac k krát. Veľkosť takéhoto programu budeme označovať $BP_k(P)$ (veľkosť najmenšieho read k-times only branching programu pre funkciu f ako $BP_k(f)$). Read 1-time only BP sa označujú ako read-once BP.

Rozlišuje sa medzi sémantickým read k-times only BP, vtedy sa obmedzenie týka ciest, ktoré môžu byť výpočtami a syntaktickým, kedy sa obmedzenie na čítanie vstupu týka všetkých orientovaných ciest (aj takých, ktoré nie sú dosiahnuteľné). Pokiaľ nenapíšeme inak pod BP_k myslíme syntaktický. V BP_1 ku každej ceste od koreňa k niektorému listu existuje vstup, ktorý túto cestu aktivuje. Dôvod je ten, že na nedosiahnuteľnej ceste sa musí testovať premenná aspoň dva krát, čo sa na BP_1 nedá. Preto medzi syntaktickým a sémantickým BP_1 nie je rozdiel. Väčšinou sa študovali syntaktické BP_k, aj keď prirodzenejšie sú sémantické. Nedosiahnuteľné cesty nás nemusia zaujímať, veď neexistuje výpočet ktorý by takú cestu sledoval. Dlho bola otvorená otázka, či sú triedy polynomiálnych syntaktických a sémantických BP_k ekvivalentné. Odpoveď priniesli Beame, Saks a Thathachar (12). Ukázali funkcie, ktoré sú na sémantických BP_2 ľahké (polynomiálne), ale na syntaktických BP_k ostávajú ťažké (exponenciálne).

BP_k je vlastne model s ohraničenou hĺbkou, keďže hĺbka BP_k nemôže byť viacej ako $k \cdot n$. Tým ale nie je obmedzená jeho výpočtová sila, nakoľko aj BP_1 dokážu definovať všetky jazyky (program z lemy 2.1.1 je read-once).

Symbolom $DT(P)$ označíme veľkosť BP P, ktorého graf je strom (rozhodovací strom). Hierarchia je definovaná nasledovne.

Definícia 3.2.2: (13) Nech C je nejaká zložitostná miera pre booleovské funkcie. Potom $C(P)$ je trieda všetkých postupností booleovských funkcií $f_n \in B_n$ takých, že $C(f_n) \leq p(n)$ pre nejaký polynóm p.

Zrejme platia vzťahy

$$DT(P) \subseteq BP_1(P) \subseteq \cdots \subseteq BP_k(P) \subseteq BP_{k+1}(P) \subseteq \cdots \subseteq BP(P)$$

Teda to čo sa dá definovať pomocou BP_k s polynomiálnou veľkosťou sa dá aj na BP_{k+1}. Úplne prvá inklúzia je zrejmá z faktu, že BP s grafom, ktorý je strom, nemá prečo testovať nejakú premennú viackrát. Keď sa výpočet dostane do vrcholu v a predtým sa testovala premenná x_i, v podstrome s koreňom v je známa hodnota premennej x_i.

Platí vzťah $DT(P) \subset BP_1(P)$, napríklad funkcia parity 1 na vstupe sa dá na BP_1 s veľkosťou $O(n)$, ale na rozhodovacích stromoch potrebuje až exponenciálnu veľkosť.

Zaujímavšiu vlastnú inklúziu dokázal Wegener (13). Podarilo sa mu ukázal, že $BP_1(P) \subset BP_2(P)$. Použil nato funkciu „exactly half clique“.

Definícia 3.2.3: (13) Exactly half clique $ECLIQUE_n$ je funkcia všetkých neorientovaných grafov s počtom vrcholov n, ktoré obsahujú kompletný podgraf o $\lceil n/2 \rceil$ vrcholoch (half-kliku) a zvyšných $\lfloor n/2 \rfloor$ vrcholov je izolovaných. Vstupom je $\binom{n}{2}$ bitov $x_{i,j}$ ($1 \leq i < j \leq n$), ktoré tvoria maticu susedností grafu ($x_{i,j} = 1$ vtedy a len vtedy, keď medzi vrcholmi i a j je hrana).

Wegener v závere svojej práce (13) vyslovil domnienku, že pre všetky $k \geq 1$ platí vzťah $BP_k(P) \subset BP_{k+1}(P)$. Uviedol aj kandidátov na odseparovanie jednotlivých tried. Úplná separácia sa podarila až Thathacharovi (14). Ukázal funkcie (pre každé k), ktorým stačí na definovanie BP_{k+1} s lineárnou veľkosťou, ale BP_k na ich reprezentáciu potrebuje veľkosť aspoň $2^{\Omega(n^{1/k+1}2^{-2k}k^{-4})}$, čo je exponenciálne veľa.

V súvislosti s BP_k sa často používajú aj nedeterministické BP. Tie si, podobne ako iné nedeterministické modely, môžu počas výpočtu „tipnúť“ ako budú pokračovať.

Definícia 3.2.4: (15) Nedeterministický BP (NBP) obsahuje aj vrcholy s dvoma výstupnými hranami, v ktorých sa nečíta žiadna premenná zo vstupu. V nich výpočet pokračuje niektorou z týchto 2 hrán (nedeterministicky sa uhádne). Funkcia reprezentovaná $NBP\ P$ vracia na vstupe a hodnotu 1, ak v P existuje výpočet, ktorý skončí v liste s označeným 1.

Používa sa aj definícia, v ktorej majú vrcholy ľubovoľný počet výstupných hrán, z nich niektoré majú označenie 1, ostatné majú 0. Ľahko vidieť že takáto definícia je ekvivalentná z vyššie uvedenou. Nedeterministické read k-times only BP (NBP_k) sú potom definované podobne ako v definícií 3.2.1.

3.3 Read $(1, +k)$-branching programy

Ďalším možným zovšeobecnením BP_k sú programy, ktorým povolíme niektoré premenné čítať viackrát. Ako uvidíme v kapitole o dolných odhadoch, na BP_1 sa podarilo dokázať rôzne netriviálne tvrdenia o ich veľkosti. Zároveň BP_k ($k \geq 2$) dlho úspešne odolávali akýmkoľvek pokusom o získanie netriviálnych odhadov pre ich zložitosť. Preto sa medzera medzi BP_1 a BP_2 rozbila ma menšie pomocou $(1, +k)$ −branching programov, kde najviac k premenných dovolíme čítať viackrát.

Definícia 3.3.1: BP P sa nazýva read $(1, +k)$, ak počet premenných, ktoré sa testujú na ceste z koreňa do niektorého listu viac ako raz nepresiahne k.

Podobne ako pri BP_k rozlišujeme medzi syntaktickými a sémantickými obmedzeniami. Rovnako môžeme zaviesť aj nedeterminizmus do výpočtov.

Trieda polynomiálnych $(1, +k) - BP$ sa ukázala byť slabšia ako polynomiálnych $\big(1, +(k+1)\big) - BP$. A to aj pre všeobecný (sémantický) prípad. Tento výsledok dosiahli Savický a Žák (16).

Kapitola 4

Dolné odhady zložitosti branching programov

Jedným z najťažších a zároveň najdôležitejším problémov zložitosti je určovanie dolných odhadov pre explicitne definované jazyky (pri BP postupností funkcií). Horné odhady sa hľadajú relatívne ľahko, hovoria o tom, že existuje inštancia BP, ktorá reprezentuje zadanú funkciu s konkrétnou zložitosťou. Nato stačí program nájsť a ukázať, že mu stačia obmedzené prostriedky. Naproti tomu pri dolných odhadov, potrebujeme ukázať neexistenciu BP, ktorý by definoval funkciu so zložitosťou menšou. Nie je problém ukázať niektoré slabé dolné hranice na zložitosť. Ale už napríklad kvadratické hranice sú veľmi ťažké. Práve veľké dolné odhady nás zaujímajú, bez nich sa nedá dokázať, že nejaký BP je optimálny. Keď sa horný a dolný odhad rovnajú, získali sme asymptoticky presný odhad zložitosti problému. Pokiaľ sa ale nerovnajú, stále sa môže stať, že na popísanie funkcie existuje aj efektívnejší BP ako poznáme.

Prvý problém je samotná definícia explicitne definovanej funkcie. Ak každej funkcií na n premenných priradíme nejaký kód (napríklad pravdivostnú tabuľku) a usporiadame ich lexikograficky podľa tohto kódu. Môžeme potom definovať postupnosť funkcií nasledovne. f_i bude prvá funkcia na i premenných, pri ktorej BP potrebuje na jej realizáciu veľkosť aspoň i^3. Získať dolný odhad na takejto postupnosti nie je žiadny problém, ale inak o nej veľa nevieme. Preto sa takto určená postupnosť nepovažuje za explicitne definovanú. Pokiaľ sa vyhneme podobným trikom pri popise postupností, tak skoro všetko prakticky používáne funkcie sú explicitne definované. Formálne (1), postupnosť $\{f_i\}_{i=0}^{\infty}$ môžeme chápať ako explicitne definovanú ak $\bigcup_{i=0}^{\infty}\{x_1 \dots x_i | f_i(x_1, \dots, x_i) = 1\} \in NP$.

Najjednoduchší horný odhad nám dáva dôkaz lemy 2.1.1. BP pre funkciu f nad n premennými skonštruovaný podľa postupu z dôkazu bude mať tvar úplného binárneho stromu s hĺbkou n. Čiže každá booleovská funkcia sa dá ma modeli BP vyhodnotiť v $O(n)$. Takýto strom má ale až $2^n - 1$ vnútorných vrcholov, preto na každú funkciu stačí BP s veľkosťou $O(2^n)$. Získali sme síce lineárnu hĺbku na úkor exponenciálnej veľkosti. Teraz ukážeme, že pre skoro všetky funkcie potrebujeme exponenciálne veľký BP.

Lema 4.0.1: Najviac $s \cdot n^s \cdot (s+1)^{2s}/s!$ rôznych funkcií $f \in B_n$ môže byť reprezentovaných BP s veľkosťou s.

Dôkaz: Uvažujeme striktnú definíciu BP (graf, ktorý má len dva listy, jeden pre 1 a druhý pre 0). To určite nezmenší silu BP. Graf pre BP má s vnútorných vrcholov, každý z nich môže byť označený jednou premennou z n možných. Výstupné hrany môžu ísť do jedného z $s-1$ vnútorných vrcholov, alebo do niektorého listu. Koreň môžeme zvoliť ľubovoľne z s vrcholov. To nám zatiaľ dokopy dáva $s \cdot n^s \cdot (s+1)^{2s}$ možností pre označenia vrcholov a ich vzájomných poprepájaní. Obsahuje to aj syntakticky nekorektné programy, napríklad sú tam cyklické grafy. To nijak nevadí lebo robíme odhad zhora. V skutočnosti rôznych funkcií bude oveľa menej. Permutácií vrcholov je $s!$, každá permutácia (ak je syntakticky správna) realizuje tú istú booleovskú funkcie, lebo výsledok nezávisí od usporiadania vrcholov, ale iba od ich označení. Čiže tvrdenie lemy platí. ■

Veta 4.0.1: (6) Pre skoro všetky[7] booleovské funkcie $f \in B_n$ platí, že BP na ich realizáciu má veľkosť aspoň $s(n) = \frac{2^n}{n}\left(1 - \frac{1}{\sqrt{n}}\right)$.

Dôkaz: Využije sa fakt, že všetkých booleovských funkcií na n premenných je 2^{2^n}. Zoberie sa z toho časť funkcií s počtom $2^{2^n - 2^{n/2}}$ a ukáže sa, že aspoň jedna z týchto funkcií nie je reprezentovateľná BP s veľkosťou $s(n)$. To sa podarí na základe lemy 4.0.1, kde stačí ukázať

$$s \cdot n^s \cdot (s+1)^{2s}/s! < 2^{2^n - 2^{n/2}}, \text{ kde } s = \frac{2^n}{n}\left(1 - \frac{1}{\sqrt{n}}\right)$$

pre nejaké veľké n. ■

Táto veta hovorí o tom, že skoro všetky booleovské funkcie sú veľmi ťažké. Potrebujú na svoju reprezentáciu BP s exponenciálnou veľkosťou. Ak teda zoberieme funkciu f takú, že pre každý vstup a zvolíme $f(a) = 1$ s pravdepodobnosťou $1/2$, môžeme pre ňu očakávať BP s komplikovanou štruktúrou. Nič ale nehovorí o odhadoch explicitne definovaných funkcií. Tie sú spravidla ľahšie, lebo majú predvídateľnú štruktúru. Funkčné hodnoty sa nevyberajú náhodne, ale na základe istých vlastností vstupu, preto stačí overiť v BP tieto vlastnosti, čo ich robí menšími.

Môžeme sa pokúsiť získať dolné odhady na BP pomocou odhadov na inom modely. Jednoduchá cesta sa zdajú byť booleovské obvody (definované v 2.2), keďže sme ich už viackrát spomínali ako model veľmi blízky BP.

Lema 4.0.2: Pre ľubovoľnú booleovskú funkciu f platí $C(f) \leq 3 \cdot BP(f)$.

Dôkaz tejto lemy je jednoduchý, stačí sa na výpočet BP pozrieť v zmysle definície 1.1.4 a hneď je vidieť, že jeden vrchol BP sa dá simulovať po pomocou 3 booleovských

[7] Všetky, až na exponenciálne malú časť.

spojok. Podľa tohto tvrdenia, ak máme dobré dolné hranice pre booleovské obvody máme asymptoticky také isté hranice aj pre BP. Problém je ale nájsť väčšie ako lineárne hranice na obvodoch. Preto nám táto lema veľmi nepomáha.

4.1 Nečiporukova metóda

Nečiporuk v roku 1966 publikoval svoju prácu (17), v ktorej uvádza metódu pre tvorbu dolných odhadov zložitosti explicitne definovaných funkcií na booleovských obvodov. Táto metóda sa dá úspešne previesť do reči BP. Založená je na intuitívnom pozorovaní, že funkcia, ktorá má veľa podfunkcií (čo implikuje zložitú štruktúru) nemôže mať príliš malú (jednoduchú) veľkosť obvodu (BP).

Definícia 4.1.1: Nech f je booleovská funkcia definovaná na množine premenných X_n. Funkciu f' nazveme podfunkciou funkcie f, ak f' vznikla z f dosadením konštánt za niektoré premenné z X_n.

Napríklad funkcia $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee \overline{x_3}$ má ako podfunkciu $f'(x_1, x_2) = (x_1 \wedge x_2)$ ak za x_3 dosadíme 1.

Definícia 4.1.2: (6) Nech S je podmnožina množiny premenných (vstupov) X_n a f je booleovská funkcia, potom každú podfunkciu, ktorá vznikne z f substitúciou konštánt za premenné z množiny $X_n - S$ nazveme S-podfunkciou.

Ak zoberieme, za množinu $S = \{x_2, x_3\}$, potom funkcia f z príkladu vyššie má nasledujúce S-podfunkcie. $f_1 = \overline{x_3}$ (ak za x_1 dosadíme 0) a $f_2 = x_2 \vee \overline{x_3}$ (ak za x_1 dosadíme 1).

Veta 4.1.1: (6) Nech X_n je množina premenných, $f \in B_n$ je booleovská funkcia definovaná na X_n, ktorá závisí od všetkých svojich premenných. Nech $S_1, \dots, S_k \subseteq X_n$ sú disjunktné podmnožiny X_n a nech s_i označuje počet rôznych S_i-podfunkcií f, potom platí

$$BP(f) = \Omega\left(\sum_{i=1}^{k} \frac{\log s_i}{\log \log s_i}\right)$$

Dôkaz: Počet rôznych BP o veľkosti t realizujúcich funkcie na r premenných je podľa lemy 4.0.1 zhora ohraničený vzťahom

$$t \cdot r^t \cdot (t+1)^{2t}/t! \leq r^t \cdot t^{2t}, \text{ pre } t \geq 4 \qquad \text{(i)}$$

Označme r_i mohutnosť množiny S_i a t_i počet vrcholov v optimálnom BP pre f označených nejakou premennou $x \in S_i$. Každá S_i-podfunkcia f môže byť reprezentovaná BP o veľkosti t_i. Stačí ak premenné mimo S_i nahradíme za konštanty. V grafe BP to vykonáme takto. Ak vrchol v má byť nahradený za konštantu b, zoberieme jeho predchodcov, označme ich ako $u_1, \dots, u_l$. Pre každého predchodcu u_i nech ide do v hrana b_i. Potom vrchol, ktorý je nasledovníkom v po hrane prislúchajúcej konštante b pripojíme na každý u_i cez hranu b_i. Do v nevedie už žiadna

cesta, čiže jeho ohodnotenie nijak neovplyvňuje výpočet. Tým sme dostali BP v ktorom je za v dosadená konštanta b a znížili sme jeho veľkosť o 1. Po vykonaní všetkých substitúcií určite dostaneme BP s veľkosťou t_i.

Podľa odhadu (i) sa preto počet S_i-podfunkcií dá odhadnúť

$$s_i \leq t_i \cdot {r_i}^{t_i} \cdot (t_i + 1)^{2t_i}/t_i! \leq {r_i}^{t_i} \cdot {t_i}^{2t_i}, \text{ pre } t_i \geq 4 \qquad \text{(ii)}$$

Funkcia f závisí od všetkých premenných, preto sa určite každá aspoň raz vyskytuje ako ohodnotenie vrcholu a teda $r_i \leq t_i$. Vzťah (ii) sa dá odhadnúť ako

$$s_i \leq {t_i}^{3t_i} \qquad \text{(iii)}$$

Sporom ukážeme odhad pre $t_i = \Omega(\log s_i / \log\log s_i)$. Nech teda pre ľubovoľnú konštantu c platí $t_i < c(\log s_i / \log\log s_i)$. Zvolíme konštantu $c = 1/3$ z platnosti (iii) dostávame vzťah

$$s_i < (c \cdot \log s_i / \log\log s_i)^{3c(\log s_i / \log\log s_i)}$$

použijeme substitúciu $a = \log s_i$ a upravíme

$$\begin{aligned}
2^a &< \left(\frac{a}{3 \cdot \log a}\right)^{(a/\log a)} \\
\log 2^a &< \log\left(\left(\frac{a}{3 \cdot \log a}\right)^{(a/\log a)}\right) \\
a &< \left(\frac{a}{\log a}\right) \cdot \log\frac{a}{3 \cdot \log a} \\
\log a &< \log a - \log\log a - \log 3 \\
\log\log a &< -\log 3 \\
\log\log\log s_i &< -\log 3
\end{aligned}$$

Posledná nerovnosť určite neplatí, lebo s_i je kladné. Čiže odhad pre t_i platí.

Keďže veľkosť optimálneho BP pre f je zjavne súčet všetkých t_i dostávame tvrdenie vety

$$BP(f) = \sum_{i=1}^{k} t_i = \Omega\left(\sum_{i=1}^{k} \frac{\log s_i}{\log\log s_i}\right)$$

■

Postup pri získavaní dolných hraníc pomocou Nečiporukovej metódy je:

3) Pokúsime sa nájsť podmnožiny S_i premenných, ktoré nám vytvoria podfunkcie f.
4) Spočítame čísla s_i (počet S_i-podfunkcií) tak, že odhadneme počet priradení premenným z $X_n - S_i$, ktoré vedú k rôznym podfunkciám.
5) Pomocou vety 4.1.1 určíme dolnú hranicu.

Metódu budeme ilustrovať na probléme $3 - CLIQUE_n$.

Definícia 4.1.3: (13) Funkcia $k - CLIQUE_n$ je funkcia, ktorej vstupom je $\binom{n}{2}$ bitov $x_{i,j}$ $(1 \leq i < j \leq n)$, ktoré tvoria maticu susedností grafu s n vrcholmi ($x_{i,j} = 1$ vtedy a len vtedy, keď medzi vrcholmi i a j je hrana). Výsledkom funkcie ma vstupe x je 1 práve vtedy, keď graf G definovaný vstupom x obsahuje kompletný podgraf s k vrcholoch (k-kliku).

$3 - CLIQUE$ je jazyk všetkých neorientovaných grafov s n vrcholoch, ktoré majú trojuholník (K_3). Horná hranica pre tento problém sa dá na BP získať celkom jednoducho.

Veta 4.1.2: $BP(3 - CLIQUE_n) = O(n^3)$.

Dôkaz: Problém hľadania 3-kliky sa dá ekvivalentne prepísať do splniteľnosti booleovských formúl ako $A(n) = \bigvee_{1 \leq i < j < k \leq n} (x_{i,j} x_{j,k} x_{i,k})$. Je to formula v disjunktívnej normálnej forme, kde klauzuly sú konjunkcie troch premenných, ktoré reprezentujú hrany, medzi nejakou trojicou vrcholov. Ak je klauzula splnená, potom určite tieto vrcholy tvoria trojuholník. Takto sú tam v disjunkcií všetky možné trojice, čím je zabezpečená ekvivalencia splniteľnosti tejto formule $A(n)$ a existencie 3-kliky v zodpovedajúcom grafe na n vrcholoch. Pre konjunkciu 3 premenných podľa lemy 2.1.1 určite existuje BP o veľkosti d.

Lema 4.1.1: Disjunkcia dvoch funkcií, ku ktorým existujú BP P_1 a P_2 o veľkosti k a l sa dá realizovať pomocou BP s veľkosťou $k + l$.

Dôkaz: Zoberieme P_1 a P_2 v tvare s 2 listami (zjednotíme všetky listy s 1 do jedného vrcholu, čím nijak nezmeníme sémantický význam programu, podobne pre 0). Výsledný BP pre disjunkciu získame tak, že koreň P_2 dáme namiesto listu s označením 0 v programe P_1. Takto vzniknutý BP bude mať veľkosť $k + l$ a ľahko vidieť, že realizuje disjunkciu. ■

Počet klauzúl v $A(n)$ je $\binom{n}{3}$, preto počet disjunkcií v $A(n)$ je $\binom{n}{3} - 1$. Jedna disjunkcia dvoch BP s veľkosťami d sa dá podľa lemy 4.1.1 realizovať BP s veľkosťou $2d$. Preto sa $\binom{n}{3} - 1$ disjunkcií dá realizovať BP s veľkosťou $\binom{n}{3} d = O(n^3)$. ■

Veta 4.1.3: (18) $BP(3 - CLIQUE_n) = \Omega(n^3 / \log n)$.

Dôkaz: Budeme postupovať podľa Nečiporukovej metódy. Za množiny S_i zvolíme $S_i = \{x_{i,i+1}, x_{i,i+2}, \dots, x_{i,n}\}$ pre $i = 1, \dots, n-1$. Ako podfunkcie zoberieme 3-kliku na grafoch s $n - i + 1$ vrcholmi. Preto pri určovaní čísla s_i zafixujeme všetky premenné $x_{k,l} = 0$ pre $k = \{1, \dots, i\}$ a $l = \{1, \dots, n\}$, alebo $k = \{1, \dots, n\}$ a $l = \{1, \dots, i\}$. Teda všetky hrany, ktoré by mali byť incidentné s vrcholom z množiny $\{1, \dots, i-1\}$ nastavíme na 0. Pre množinu S_i máme graf G_i, ktorý má ako vrcholy $\{i, \dots, n\}$. Množina S_i obsahuje premenné prislúchajúce k hranám, ktoré sú susedné z vrcholom i. Aby sme naplnili definíciu podfunkcie a vyjadrili s_i, musíme ešte určiť počet priradený zvyšným hranám v G_i, ktoré budú viesť k rôznym podfunkciám. Priradíme im také hodnoty, aby

vytvárali spolu bipartitný graf s rovnomerne rozdelenými partíciami $\{i+1,\dots,i+\lceil(n-i)/2\rceil\}$ a $\{i+\lceil(n-i)/2\rceil+1,\dots,n\}$. Takýchto priradení máme na výber $2^{\lfloor(n-i)/2\rfloor\lceil(n-i)/2\rceil}$. Akoby sme vyplnili tabuľku susedností iba na jednej štvrtine, kde sú hrany medzi našimi partíciami. Zvyšným premenným (hrany v rámci partícií) priradíme hodnotu 0. Bipartitný graf sme volili preto, lebo neobsahuje trojuholníky (keďže je bipartitným nemôže obsahovať kružnice nepárnej dĺžky).

Ďalej musíme ukázať, že k rôznym priradeniam prislúchajú rôzne podfunkcie. Zoberme 2 rôzne priradenia a a b. Bez ujmy na všeobecnosti nech sa líšia v premennej $x_{j,k}$ a nech v priradení a sa $x_{j,k}=1$. Potom je podfunkcia f_a na množine S_i pre priradenie a rozdielna od podfunkcie f_b pre priradenie b. Keď premenné z S_i nastavíme nasledovne $x_{i,p}=1$ pre $p\in\{j,k\}$ a $x_{i,p}=0$ inak. Pre priradenie a existuje v grafe G_i trojuholník $\{j,k,i\}$ a teda výsledok $f_a=1$ a pre b určite v G_i trojuholník neexistuje ($f_b=0$). Pre dve rôzne priradenia sú S_i-podfunkcie rôzne, preto číslo s_i sa dá zdola odhadnúť $s_i\geq 2^{\lfloor(n-i)/2\rfloor\lceil(n-i)/2\rceil}$.

Na záver, keď už máme vypočítane čísla s_i, môžeme odhadnúť výraz z vety 4.1.1.

$$\log s_i \geq \lfloor(n-i)/2\rfloor\lceil(n-i)/2\rceil = \frac{(n-i)^2}{4} - O(n)$$

$$\log\log s_i \approx 2\log n$$

$$BP(3-CLIQUE_n) = \Omega\left(\sum_{i=1}^{n-1}\frac{\log s_i}{\log\log s_i}\right) = \Omega\left(\sum_{i=1}^{n-1}\frac{(n-i)^2-O(n)}{8\log n}\right) =$$

$$= \Omega\left(\frac{1}{8\log n}\sum_{i=1}^{n-1}\left((n-i)^2-O(n)\right)\right)$$

Suma $\sum_{i=1}^{n-1}(n-i)^2 = \sum_{i=1}^{n-1} n^2 = n^3/3 + O(n^2)$ a teda pre dolný odhad platí

$$BP(3-CLIQUE_n) = \Omega\left(\frac{1}{8\log n}\left(\frac{n^3}{3}+O(n^2)\right)\right) = \Omega\left(\frac{n^3}{\log n}\right)$$

■

Rozdiel medzi horným a dolným odhadom pre funkciu $3-CLIQUE_n$ je malý, líšia sa len o logaritmický faktor. Algoritmus z vety 4.1.2 teda nemusí byť optimálny a stále sa môžeme pokúšať hľadať lepší, ktorý dosahuje dolnú hranicu. Alebo môžeme vylepšovať dolný odhad až na $\Omega(n^3)$, čím by sa algoritmus ukázal ako optimálny.

Pomocou Nečiporukovej metódy sa nedajú získať odhady väčšie ako $\Omega\left(\frac{n^2}{\log^2 n}\right)$ (6), kde n je počet vstupov (vo vete 4.1.3 je n počet vrcholov grafu a nie počet vstupov). Zároveň existujú funkcie, ktoré túto hranicu, použitím tejto metódy, dosahujú. Takou funkciou je napríklad funkcia „odlišnosti prvkov“.

Definícia 4.1.4: Funkcia $Distinct_m$ je booleovská funkcia, ktorej vstupom je m-tica čísel z intervalu 1 až m^2, každé zapísané binárne pomocou $2\log m$ bitov. Pre vstup a platí, že $Distinct_m(a) = 1$ práve vtedy, keď všetkých m čísel zakódovaných v a je navzájom rôznych.

Veta 4.1.4: Znova použijeme Nečiporukovu metódu. Za množinu S_i zvolíme premenné na vstupe, ktoré zodpovedajú číslu i ($1 \leq i \leq m$). Aby sme určili číslo s_i musíme určiť počet podfunkcií, ktoré vzniknú po dosadení rôznych konštánt za premenné mimo S_i. Vyberieme tie priradenia kde sú všetky čísla rozdielne. Tých je $\binom{m^2}{m-1}$, pre dve rôzne priradenia sú aj podfunkcie na nich definované rôzne. Priradenia sa určite líšia aspoň v jednom čísle. Ak za S_i zvolíme práve reprezentáciu tohto čísla, tak v jednom priradení bude výsledok funkcie, na ňom definovanej, 1 a v druhom 0. Teraz odhadneme výrazy $\log s_i$ a $\log\log s_i$.

$$\log s_i = \log\binom{m^2}{m-1} = \log\frac{m^2 \cdot \ldots \cdot (m^2 - m + 1)}{(m-1) \cdot \ldots \cdot 2 \cdot 1}$$
$$\geq \log\left(m^2 \cdot \ldots \cdot (m^2 - m/2 - 1)\right) \geq \log m^{m/2}$$

Predposledná nerovnosť vychádza z faktu, že $(m-1)(m-2) \leq (m^2 - m + 1)$ a teda súčin každej dvojice z menovateľa v predchádzajúcom výraze je menší ako ľubovoľný činiteľ z čitateľa.

$$\log\log s_i = \log\log\binom{m^2}{m-1} \leq \log\log m^{2m^2}$$

Odhad získame z vety 4.1.1, pre veľké m platí

$$\frac{\log s_i}{\log\log s_i} \geq \frac{\log m^{m/2}}{\log\log m^{2m^2}} \geq \frac{m\log m}{2\log(2m^2\log m)} \geq \frac{m\log m}{2\log(m^3)} \geq \frac{m}{6}$$

$$BP(Distinct_m) = \Omega\left(\sum_{i=1}^{m}\frac{\log s_i}{\log\log s_i}\right) = \Omega\left(\sum_{i=1}^{m}\frac{m}{6}\right) = \Omega(m^2)$$

Veľkosť vstupu funkcie $Distinct_m$ je $n = 2m\log m$. Teda sme dostali odhad

$$BP(Distinct_m) = \Omega\left(\frac{n^2}{\log^2 n}\right)$$

■

Pre explicitne definované funkcie na BP bez ohraničení doteraz neexistuje iná technika ako získavať netriviálne dolné odhady. Preto sa ďalej pozrieme na isté ohraničené triedy. Snaha je pomocou výskumu v oblastí hraníc na ohraničených BP nazbierať dostatok vedomostí, aby sa dalo zaútočiť na všeobecný prípad.

4.2 Branching programy so šírkou dva

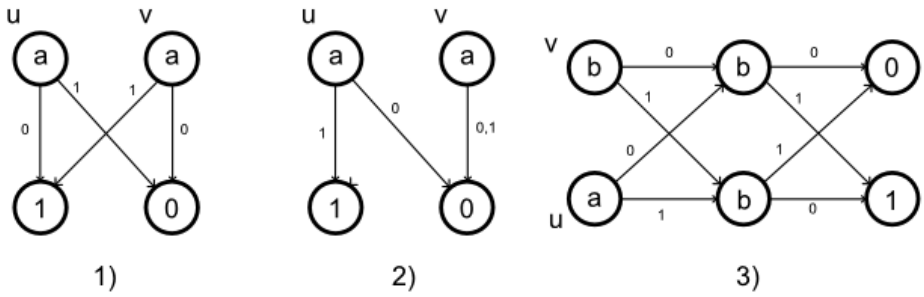
Ak šírku obmedzíme na dva a pritom nedáme žiadne ohraničenie na ostatné zdroje BP, potom podľa lemy 3.1.1 vieme definovať ľubovoľný jazyk. Používať BP so šírkou jedna nemá príliš zmysel. Takýto BP zvládne iba konštantné funkcie, lebo existuje v ňom iba jedna cesta. Prirodzené je ale začať skúmať BP so šírkou dva. V tejto časti sa budeme opierať o prácu Borodina, Doleva, Ficha a Paula (19), ktorí sa prví zaoberali takto ohraničenými BP z hľadiska dolných odhadov zložitosti. Z nej čerpáme aj tvrdenia v tejto kapitole.

Podobne ako vo vete 4.0.1 by sme dospeli aj k tvrdeniu, že skoro všetky funkcie musia mať v triede W_2 BP s exponenciálnou veľkosťou. Zaujímajú nás hlavne odhady pre explicitne definované funkcie.

V (19) sa používa jemnejšie delenie BP so šírkou 2 na monotónne (zamietacie vrcholy sú povolené až na poslednej úrovni) a striktné (akceptačný a zamietací vrchol môže byť iba na poslednej úrovni). Trieda striktných BP so šírkou 2 sa označuje ako SW_2. Každý všeobecný BP so šírkou dva s k akceptačnými alebo zamietacími vrcholmi môže byť prirodzenou cestou rozložený do najviac k striktných BP. Vieme, že monotónne W_2 BP stačia na definovanie každej booleovskej funkcie (BP z dôkazu k leme 3.1.1 bol vlastne monotónny), nevieme zatiaľ akú triedu funkcií rozpoznávajú BP z SW_2. Dva vrcholy na jednej vrstve v striktných BP zodpovedajú akoby dvom rôznym stavom. Keďže BP si môže pamätať nejakú informáciu iba tým, v ktorom vrchole sa výpočet nachádza. Striktné nemajú možnosť priebežne akceptovať, preto trieda SW_2 neobsahuje všetky booleovské funkcie. O jej presnej sile hovorí nasledujúca veta.

Veta 4.2.1: (19) SW_2 je najmenšia trieda booleovských funkcií S, ktorá obsahuje konštantné funkcie 0, 1, projekcie (funkcie $g(x_1, \dots, x_n) = x_i$, pre všetky n a i) a ak $f \in S$ a a, b sú literály

1) $f \oplus a \in S$
2) $f \wedge a \in S$
3) $f \wedge (a \oplus b) \in S$



Obr. 6: Príklady vytvorenia BP pre rozdielne prípady z vety 4.2.1. Na obrázku je symbolom a označená premenná prislúchajúca k literálu a. Ak a je negatívny literál, potom sa prehodia hrany idúce od vrcholu a. Podobne aj pre b.

Dôkaz: Jedna inklúzia ($S \subseteq SW_2$) sa ukáže ľahko. Konštantné funkcie a projekcie sa zrejme dajú realizovať na striktných BP. Nech $f \in SW_2$, potom k f existuje striktný BP P, ktorý realizuje f. Ďalej nech u je akceptačný vrchol P a v zamietací. Ako sa vytvoria BP pre rôzne prípady z vety je uvedené na Obr. 6.

Ostáva ukázať opačnú inklúziu ($SW_2 \subseteq S$). Budeme postupovať indukciou na dĺžku programu. Báza indukcie je zrejmá. V S sú určite všetky funkcie realizované BP s dĺžkou najviac 1 (to sú konštanty a projekcie). Predpokladajme, že v S sú všetky funkcie, ktoré majú BP s dĺžkou najviac n. Uvažujme funkciu f, ktorá ma BP P o veľkosti $n+1$. Pozrieme sa na posledné dve úrovne. Na poslednej úrovni máme vrchol s označeným 1 a vrchol s 0. Označme vrcholy predposlednej vrstve ako u a v (podobne ako na Obr. 6 časti 1)). Funkcia g reprezentovaná BP P, ktorému sme odobrali poslednú vrstvu a vrcholy u a v sme položili za akceptačný a zamietací, patrí podľa indukčného predpokladu do S. Označme ako f_u funkciu, ktorá zodpovedá BP P, ak začneme výpočet vo vrchole u. Funkciu f_v definujeme analogicky. Nech x_i je premenná testovaná v u, potom triviálne platí, že f_u je jedna z funkcií $0, 1, x_i, \overline{x_i}$, podobne aj pre f_v. Potom funkcia f reprezentovaná BP P sa dá napísať ako $f = \left(g \wedge \left(\overline{f_u} \oplus \overline{f_v}\right)\right) \oplus f_v$. Ak výsledok g je 0, potom podľa predpisu pre f je výsledkom f_v, čo zodpovedá štruktúre P. Naopak ak $g = 1$, potom je výsledok $f = f_u$, čo znova zodpovedá konštrukcii. Takáto funkcia f je v S, lebo sa dá vyskladať pomocou vlastností 1) a 3). Všimnime si, že pravidlá 1) a 3) rátajú len s literálmi a nie s konštantami. Preto ak f_u alebo f_v je konštantná funkcia nemôžeme ich aplikovať. Rozobratím prípadov pre jednotlivé priradenia konštantných funkcií funkciám f_u alebo f_v by sme dospeli k výsledku, že f sa vždy dá vyskladať pomocou pravidiel z lemy. ■

V SW_2 sú funkcie ako parita 1 na vstupe ($x_1 \oplus \ldots \oplus x_n$), funkcia, ktorá hovorí či sú na vstupe samé 1 ($x_1 \wedge \ldots \wedge x_n$). Všetky tieto funkcie vznikli používaním pravidiel 1), 2) a 3) z predchádzajúcej vety. Ako sme už skôr naznačili, nie všetky funkcie patria do SW_2. Ktoré to sú popisuje nasledujúca lema.

Lema 4.2.2: (19) Nech $f \in SW_2$ potom platí jedno z nasledujúcich tvrdení:

1) f je konštantná, alebo paritná[8] funkcia
2) existuje literál a taký, že po substitúcií $a = 0$, dostaneme z f konštantnú, alebo paritnú funkciu
3) existujú dva literály a a b také, že funkcia f sa stane konštantnou, alebo paritnou pre všetky ohodnotenia premenných, pri ktorých literály a a b nadobúdajú rovnakú hodnotu

Idea dôkazu je založená na vete 4.2.1. Kde každá funkcia z SW_2 musí byť buď konštanta, alebo paritná funkcia (vlastnosť 1)). Prípadne vznikne použitím pravidla 2) (resp. 3)) a za nim niekoľkokrát pravidlo 1). Takáto funkcia bude mať tvar $(f \wedge a) \oplus c_1 \oplus \ldots \oplus c_m$ (resp. $\big(f \wedge (a \oplus b)\big) \oplus c_1 \oplus \ldots \oplus c_m$). Čiže ak za a dosadíme 0 dostaneme tvrdenie 2) tejto lemy, podobne s tvrdením 3). Pomocou tejto lemy sa dá ukázať, že nejaká funkcia nepatrí do SW_2. Ak funkcia nespĺňa ani jednu z týchto vlastností nemôže patriť do triedy SW_2.

Ďalším pohľadom na funkcie zo SW_2, ktorým sa zaoberali autori z (19), bol geometrický pohľad. Kockou S nazveme podmnožinu $\{0,1\}^n$ tvaru

$$S = \big\{x \in \{0,1\}^n \big| x_{i_1} = a_1, x_{i_2} = a_2, \ldots, x_{i_t} = a_t \big\}, \text{ kde } a_1, \ldots, a_t \in \{0,1\}.$$

Dimenziou takejto kocky nazveme číslo $n - t$. Prúžkovanou kockou S nazveme podmnožinu $\{0,1\}^n$, ktorá spĺňa nasledujúcu vlastnosť

$$S = \Big\{x \in \{0,1\}^n \big| x_{i_1} = a_1, \ldots, x_{i_t} = a_t\,, x_{j_1} \oplus \ldots \oplus x_{j_r} = b\Big\}, \text{ kde } a_1, \ldots, a_t, b \in \{0,1\}$$

Kocky sú vlastne binárne vektory, kde zafixujeme isté premenné $(i_1, \ldots, i_t)$ na konkrétne hodnoty. Prúžkované majú naviac zafixovaný aj súčet modulo 2 niektorých premenných $(j_1, \ldots, j_r)$.

Nech Z_n označuje najmenšie číslo také, že pre každú funkciu f z SW_2 na n premenných platí – počet prúžkovaných kociek, ktorých disjunktné zjednotenie je množinu vstupov a, pre ktoré $f(a) = 1$, nepresiahne Z_n. Potom platí nasledujúca lema.

Lema 4.2.3: $Z_n \leq 4 \cdot 2^{n/2} - 2$

Dôkaz: Indukciou na parameter n. Pre $n = 1$ to zjavne platí, keďže každá funkcia na jednej premennej je buď konštanta, alebo paritná funkcia a na ne stačí jedna prúžkovaná kocka so správne nastavením súčtom premenných. Predpokladajme platnosť pre všetky čísla menšie nanajvýš rovné ako $n - 1$, ukážeme platnosť pre n. Nech $f \in SW_2$, f je definovaná na n premenných. Nech Z je najmenšie prirodzené

[8] Funkcia tvaru $a_{i_1} \oplus \ldots \oplus a_{i_m}$, kde a_{i_j} je booleovský literál.

číslo také, že množina vstupov, pre ktoré f vracia 1, je disjunktným zjednotením Z prúžkovaných kociek. Indukčný krok využíva prípady lemy 4.2.2.

1) Ak je f konštantná, alebo paritná, vtedy určite $Z \leq 1$.
2) Ak existuje literál a taký, že po substitúcií $a = 0$ sa z f stane konštanta, alebo paritná funkcia, potom platí, že $f = (a \wedge f_1) \vee (\bar{a} \wedge f_2)$, kde f_2 je príslušná konštantná, alebo paritná funkcia a $f_1, f_2 \in SW_2$ na $n-1$ premenných. Na reprezentáciu vstupov pre $(\bar{a} \wedge f_2)$ stačí určite jedna kocka a na $(a \wedge f_1)$ podľa indukčného predpokladu Z_{n-1}, dokopy teda $Z \leq Z_{n-1} + 1 \leq 4 \cdot 2^{(n-1)/2} - 2 + 1 = \left(4/\sqrt{2}\right)2^{n/2} - 1$.
3) V tomto prípade sa dá f napísať ako $f = (a \wedge b \wedge f_1) \vee \left(a \wedge \bar{b} \wedge f_2\right) \vee (\bar{a} \wedge b \wedge f_3) \vee \left(\bar{a} \wedge \bar{b} \wedge f_4\right)$, kde a, b sú literály, f_1 a f_4 sú konštanty, alebo paritné funkcie a $f_1, \dots, f_2$ sú funkcie $n-2$ premenných. Z podobných príčin ako v 2) sa dá ukázať vzťah $Z \leq 2Z_{n-2} + 2 \leq 2\left(4 \cdot 2^{(n-2)/2} - 2\right) + 2 = 4 \cdot 2^{n/2} - 2$.

Chceme ukázať, že $Z \leq 4 \cdot 2^{n/2} - 2$. Zjavne to vo všetkých prípadoch platí. ■

Po prípravných lemách sa môžeme pustiť do najzaujímavejšej časti práce (19) a to dolných odhadov na BP s výrazne ohraničenou šírkou.

Definícia 4.2.1: (19) Nech $\mathcal{S}$ je systém podmnožín $\{0,1\}^n$, $\mathcal{S}$-programom nazveme postupnosť dvojíc $(S_1, a_1), \dots, (S_m, a_m)$, kde $S_1, \dots, S_m \in \mathcal{S}$ a $a_1, \dots, a_m \in \{0,1\}$. Číslo m nazveme dĺžkou $\mathcal{S}$-programu.

Takýto $\mathcal{S}$-program počíta n-árnu funkciu f nasledovne. Pre $a \in \{0,1\}^n$, ak a $\notin \bigcup_{i=1}^{m} S_i$, potom $f(a) = 0$. Ak $a \in S_i \setminus \bigcup_{j=0}^{i-1} S_j$, potom $f(a) = a_i$. $\mathcal{S}$-zložitosť funkcie f je najmenšia dĺžka programu, ktorý realizuje f. Označíme ju ako $C_{\mathcal{S}}(f)$. $\mathcal{S}$-program $(S_1, a_1), \dots, (S_m, a_m)$ nazveme monotónny ak $\forall i: a_i = 1$. Zložitosť monotónneho $\mathcal{S}$-programu pre funkciu f označíme $MC_{\mathcal{S}}(f)$.

Zoberme funkciu $f \in W_2$, ako sme v úvode kapitoly uviedli, BP P pre f sa dá dekomponovať do postupnosti striktných BP so šírkou dva. Ak za $\mathcal{S}$ zoberieme systém všetkých prúžkovaných kociek, potom $C_{\mathcal{S}}(f)$ je počet prúžkovaných kociek. ktoré potrebujeme na reprezentáciu f pomocou $\mathcal{S}$-programu. Naviac z lemy 4.2.3 vieme, že každá funkcia z SW_2 nepotrebuje, na reprezentáciu pomocou $\mathcal{S}$-programu, viac ako $4 \cdot 2^{n/2} - 2$ prúžkovaných kociek. Preto počet programov, do ktorých sa rozloží P musí byť najmenej $C_{\mathcal{S}}(f)/\left(4 \cdot 2^{n/2} - 2\right)$. Pre hĺbku BP P (aj veľkosť, keďže tá nemôže byť viac ako dvojnásobok hĺbky) potom platí nasledujúci veta.

Veta 4.2.2: (19) Nech $f \in W_2$ a $\mathcal{S}$ je systém všetkých prúžkovaných kociek potom pre BP P realizujúci funkciu f platí:

$$BP(P) \geq BPD(P) \geq \frac{C_{\mathcal{S}}(f)}{(4 \cdot 2^{n/2} - 2)}$$

Dôkaz: Úvahy nad vetou. ■

Podobná veta platí aj pre funkcie z monotónnej W_2 a monotónne $\mathcal{S}$-programy. Ako príklad použitia nám poslúžia symetrické funkcie.

Definícia 4.2.2: Booleovská funkcia $f(x_1, \dots, x_n)$ sa nazýva symetrická, ak pre každú permutáciu π n premenných platí $f(x_1, \dots, x_n) = f\left(x_{\pi(1)}, \dots, x_{\pi(n)}\right)$.

Pri symetrických funkciách nezáleží na usporiadaní premenných, ale výsledok funkcie závisí iba od počtu 1 na vstupe.

Definícia 4.2.3: Symbolom $E_{k,l}^n$ označíme symetrickú funkciu na n premenných takú, že $E_{k,l}^n(x_1, \dots, x_n) = 1$ vtedy a len vtedy, ak $k \leq x_1 + \cdots + x_n \leq l$.

Veta 4.2.3: (19) Nech P je monotónny BP s ohraničenou šírkou dva reprezentujúci funkciu $E_{k,n}^n$, potom $BPD(P) \geq \frac{\binom{n}{k}}{n\left(4 \cdot 2^{n/2} - 2\right)}$.

Dôkaz: Pomocou vety 4.2.2. Stačí nám ukázať, že monotónny $\mathcal{S}$-program pre $E_{k,n}^n$ má veľkosť aspoň $\binom{n}{k}/n$. Zoberme jednu pruhovanú kocku S z $\mathcal{S}$-programu.

$$S = \left\{x \in \{0,1\}^n \middle| x_{i_1} = a_1, \dots, x_{i_t} = a_t\,, x_{j_1} \oplus \dots \oplus x_{j_r} = b\right\}$$

Určite aspoň $k - 1$ premenných musí byť zafixovaných na 1 pomocou a_i (posledná premenná sa môže zafixovať pomocou b). Inak by do množiny S patrili aj vstupy s počtom 1 menej ako k. Tie nepatria medzi vstupy na ktorých $E_{k,n}^n$ vracia 1 a to nemôže byť, keďže $\mathcal{S}$-program je monotónny (kde každá množina z $\mathcal{S}$ musí byť akceptačná). Preto do S nemôže patriť viac ako $n - t \leq n$ vstupov s počtom 1 presne k (z bitov mimo $\{i_1, \dots, i_t\}$ môže mať hodnotu 1 najviac ešte jeden bit). Počet vstupov, kde je práve k nastavených bitov na 1 je zrejme $\binom{n}{k}$. Preto $MC_{\mathcal{S}}\left(E_{k,n}^n\right) \geq \binom{n}{k}/n$ a teda tvrdenie vety platí. ■

Ako vidíme pre malé k je odhad triviálny, najlepšie odhady sa dosahujú pri $k \cong n/2$.

Pomocou tejto techniky sa podarilo dokázať aj ďalšie výsledky. Pre monotónne W_2 BP platí

$$BPD\left(E_{k,k}^n\right) \geq n\binom{n}{k}/(k + k^k 3^{k+1})$$

a pre všeobecné W_2 BP platí

$$BPD\left(E_{\lceil n/2 \rceil, n}^n\right) = \Omega(n^2/\log n).$$

Detailné dôkazy sa nachádzajú v (19). Na konci práce Borodin, Dolev, Fich a Paul uviedli medzi otvorenými problémami – dokázať asymptoticky väčšie ako polynomiálne dolné hranice na všeobecných W_2 BP. Z riešením prišiel Yao (20), ukázal nasledujúcu vetu.

Veta 4.2.4: Nech $p(n)$ je ľubovoľný polynóm. Potom pre veľké n BP so šírkou dva, ktorý počíta funkciu $E^n_{\lceil n/2 \rceil, n}$[9] musí mať veľkosť presahujúcu $p(n)$.

Pri dôkaze tejto vety už nebola použitá metóda spočítavania množín, ale dôkaz bol robený pravdepodobnostnými metódami. Na triede W_2 teda máme veľké odhady. Keď povolíme väčšiu šírku, už sa znova stávajú dolné odhady ťažkými. Dôvod nato je možno skrytý v Barringtonovom výsledku (kapitola 3.1), podľa ktorého sú triedy NC^1 a PBP s konštantnou šírkou ekvivalentné. Čiže BP s ohraničenou šírkou vedia rozpoznávať už celkom početnú triedu funkcií s iba polynomiálnou veľkosťou.

4.3 Odhady pre read-once branching programy

V kapitole 3.2 sme sa venovali popisu hierarchie tried read k-times only BP programov. Samozrejme aj pre tieto ohraničené modely sa začali skúmať metódy dolných odhadov. Založené sú na rôznych pozorovaniach. Jednu z techník publikoval Wegener v článku (13). Tú prezentujeme v tejto kapitole. Najprv uvedieme definície kľúčových pojmov.

Definícia 4.3.1: (13) Nech f je booleovská funkcia. Implikantom P funkcie f je konjunkcia literálov taká, že platí: ak na vstupe x je $P(x) = 1$, potom aj $f(x) = 1$.

Definícia 4.3.2: (13) Nech f je booleovská funkcia. Klauzulou C funkcie f je disjunkcia literálov taká, že platí: ak na vstupe x je $C(x) = 0$, potom aj $f(x) = 0$.

Implikant je teda konjunkcia literálov, ktorá keď je splnená na vstupe x, potom je splnená aj funkcia f. Klauzula je naopak disjunkcia literálov, ktorá keď je zamietnutá na x, potom aj f na x vracia 0. Napríklad funkcia $f(x, y, z, w) = xy \vee y\bar{z}w$ má implikanty xy, $y\bar{z}w$, ale aj xyz, $xy\bar{z}$, $xy\bar{z}w$ a ďalšie. Klauzuly sú $x \vee y$, y, $x \vee y \vee \bar{z}$ a ešte aj iné.

Definícia 4.3.3: Nech f je booleovská funkcia. Primárnym implikantom (resp. klauzulou) nazveme taký implikant (klauzulu) funkcie f, ktorý nemôže byť pokrytý všeobecnejším implikantom (klauzulou).

Teda taký implikant, z ktorého po odstránení ľubovoľnej premennej už neostane implikant funkcie. Z príkladu vyššie by to boli implikanty xy, $y\bar{z}w$ a klauzuly y, $x \vee \bar{z}$, $x \vee w$. Pod pojmom dĺžka implikantu sa myslí počet premenných v implikante (pre klauzuly obdobne).

Nasledujúca veta bude hovoriť o odhadoch pre rozhodovacie stromy (DT), je založená na pozorovaní, že ak funkcia má dlhé primárne implikanty, alebo klauzuly nemôže mať príliš malú veľkosť. Na krátkych cestách od koreňa k listom by sa nestihlo overiť či má byť výstup 1, alebo 0.

[9] Táto funkcia sa nazýva aj majority, lebo výsledok je 1 práve vtedy, ak na vstupe prevažujú bity nastavené na 1.

Veta 4.3.1: (13) Nech f je monotónna booleovská funkcia, nech $l(0)$ označuje dĺžku najkratšej primárnej klauzuly funkcie f a $l(1)$ dĺžku najkratšieho implikantu f . Potom platí:

$$DT(f) \geq \binom{l(0)+l(1)}{l(0)} - 1$$

Dôkaz: Ku každej postupnosti 0 a 1 (0-1 postupnosť) zodpovedá vrchol v grafe, do ktorého sa dostaneme z koreňa po danej postupnosti hrán (ak už dosiahneme list ďalšie čísla v postupnosti sú ignorované). Ukážeme, že postupnosti s malým počtom 1 a 0 nemôžu viesť do listu. Ku postupnosti $i_1, \dots, i_m$ označme taký vrchol ako $v(i_1, \dots, i_m)$.

Nech $i_1, \dots, i_m$ je postupnosť, kde je menej ako $l(0)$ núl a menej ako $l(0)$ jednotiek. Primárne implikanty a klauzuly monotónnej funkcie neobsahujú negované literály. Nato aby sme vedeli hodnotu funkcie f, potrebujeme mať v liste otestovaných aspoň $l(0)$ (resp. $l(1)$) premenných na hodnotu 0 (resp. 1). Na ceste $i_1, \dots, i_m$ otestujeme ale menej hodnôt ako treba na určenie výsledku funkcie f a preto výpočet ešte nemohol dôjsť až k listu. Čiže vrchol $v(i_1, \dots, i_m)$ je vnútorný v ľubovoľnom DT pre funkciu f. Veľkosť DT pre f nemôže byť menšia ako počet postupností, ktoré obsahujú menej ako $l(0)$ núl a zároveň menej ako $l(1)$ jednotiek. 0-1 postupností s k jednotkami a l nulami je zjavne $\binom{l+k}{k}$. Teda vhodných postupností je spolu

$$\sum_{i=0}^{l(0)-1} \sum_{j=0}^{l(1)-1} \binom{i+j}{i} = \binom{l(0)+l(1)}{l(0)} - 1$$

Čo je zároveň aj dolný odhad na veľkosť DT. ■

Pomocou tejto metódy sa dá ukázať dolná hranica pre funkciu $k - CLIQUE_n$ (definícia 4.1.3), funkcia je zjavne monotónna, preto môžeme použiť vetu 4.3.1.

$$DT(k - CLIQUE_n) \geq \binom{l(0)+l(1)}{l(0)} - 1$$

Kde $l(1) = \binom{k}{2}$, lebo k-klika obsahuje $\binom{k}{2}$ hrán, takže implikant musí mať aspoň toľko premenných, aby tieto hrany skontroloval. $l(0) = \left\lceil \frac{n(n/(k-1)-1)}{2} \right\rceil$, to vyplýva z Turánovej vety, ktorá hovorí o tom, že najväčší (s najväčším počtom hrán) graf na n vrcholoch, ktorý ešte neobsahuje k-kliku je taký, ktorý má $k-1$ rovnomerne rozdelených partícií $A_1, \dots, A_{k-1}$ a medzi partíciami má „všetky“ hrany. Aby sme vedeli že graf nemá k-kliku musíme skontrolovať aspoň toľko hrán, koľko je hrán v kompletnom grafe K_n mínus počet hrán v Turánovom grafe s n vrcholmi. Inak by sa nám totiž mohlo stať, že neskontrolované hrany ešte vytvárajú k-kliku. Teda $l(0) = \sum_{i=1}^{k-1} \binom{|A_i|}{2}$. Keďže $|A_i| \in \left\{ \left\lceil \frac{n}{k-1} \right\rceil, \left\lfloor \frac{n}{k-1} \right\rfloor \right\}$, po úpravách dospejeme k želanému výsledku.

Samozrejme sa dá táto technika použiť aj na iné funkcie ako hľadanie k-kliky. Dobre funguje aj na symetrických funkciách (definície 4.2.2 a 4.2.3), kde sa tiež dajú získať netriviálne odhady.

Veta 4.3.2: Pre ľubovoľnú symetrickú funkciu $E_{k,n}^n$ platí

$$DT\left(E_{k,n}^n\right) \geq \binom{n+1}{k} - 1$$

Dôkaz: Funkcia je určite monotónna, takže použijeme vetu 4.3.1. Najkratší implikant musí mať dĺžku aspoň k, lebo funkcia $E_{k,n}^n$ na vstupe s menej ako k bitmi 1, vracia hodnotu 0. Podobne pre najkratšiu klauzulu platí, že musí byť dlhá aspoň $n-k+1$. Teda platí

$$DT\left(E_{k,n}^n\right) \geq \binom{n-k+1+k}{k} - 1 = \binom{n+1}{k} - 1$$

■

Pre malé k odhady pomocou tejto vety nie sú veľmi veľké, ale ak k vzrastá s n odhady sú lepšie. Napríklad pre funkciu majority $E_{\lceil n/2 \rceil,n}^n$ dostávame odhad $DT\left(E_{\lceil n/2 \rceil,n}^n\right) \geq \binom{n}{\lceil n/2 \rceil} - 1 \geq \frac{4^{\lceil n/2 \rceil}}{\sqrt{4\lceil n/2 \rceil}} \geq \frac{2^n}{\sqrt{2n}} = \Omega\left(2^n/\sqrt{n}\right)$.

Wegener túto metódu ďalej rozvinul aj na dokazovanie dolných hraníc pre BP_1. Zavedieme označenia: $k(0)$ a $k(1)$ sú prirodzené čísla, množina $S(k)$ bude množina takých 0-1 postupností, ktoré obsahujú menej ako $k(0)$ núl a menej ako $k(1)$ jednotiek. Nech f je monotónna funkcia a P je ľubovoľný BP_1 pre f. Potom ak sa nám podarí ukázať vlastnosť, že na P pre každé dve postupnosti $(i_1, \dots, i_m), (j_1, \dots, j_p) \in S(k)$ platí $v(i_1, \dots, i_m)$ aj $v(j_1, \dots, j_p)$ nie sú listy. A naviac ak $(i_1, \dots, i_m) \neq (j_1, \dots, j_p)$ potom aj $v(i_1, \dots, i_m) \neq v(j_1, \dots, j_p)$.

Touto vlastnosťou sme vlastne ukázali, že blízko koreňa P musí byť štruktúra BP stromová. Takže môžeme rovnakým argumentom ako vo vete 4.3.1 dospieť k odhadu

$$BP_1(f) \geq \binom{k(0)+k(1)}{k(0)} - 1$$

Použitie tejto techniky je ťažšie, musíme určiť čísla $k(0)$ a $k(1)$ a ešte k tomu aj dokázať vyššie uvedenú vlastnosť množiny $S(k)$. Aj tak sa ale podarilo niečo dokázať.

Veta 4.3.3: (13) Pre funkciu $k - CLIQUE_n$ platí:

$$BP_1(k - CLIQUE_n) \geq \binom{k(0)+k(1)}{k(0)} - 1$$

Kde $k(0) = n - k^2 + 3$ a $k(1) = \binom{k}{2}$.

Ďalšiu techniku pre dolné odhady pre read-once BP ukázal Žák (21). Najprv zavedieme niektoré označenia použité v práci.

Definícia 4.3.4: Nech K je vrchol vo výpočte (postupnosť vrcholov) BP na vstupe a symbolmi a_K (resp. a^K) označíme množinu indexov vstupných bitov, ktoré sa testovali pred (resp. v a po) dosiahnutí vrcholu K.

Zjavne pre každý read-once BP a každý vrchol K platí $a^K \subseteq \overline{a_K}$ (pruhom označujeme komplement).

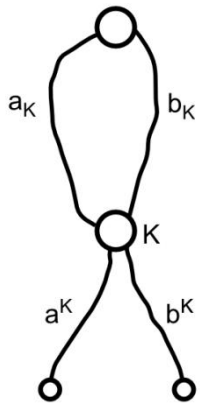
Definícia 4.3.4: (21) Nech $a = (a_1, \dots, a_n)$ a $b = (b_1, \dots, b_n)$, ďalej nech $S \subseteq \{1, \dots n\}$, potom budeme používať $a =_S b$, ak $\forall i \in S$ platí $a_i = b_i$.

Lema 4.3.1: (21) Nech P je daný BP_1, a, b sú vstupy a nech K je vrchol nachádzajúci sa vo výpočte na a aj na b. Potom platí:

1) Ak $a_K = b_K$ a c je vstup, kde $c =_{a_K} a$ a $c =_{\overline{a_K}} b$, potom c je akceptované, práve vtedy keď je akceptované b.
2) $b_K \subseteq \overline{a^K}$
3) Ak $c =_{\overline{b_K - a_K}} a$, potom výpočty na c a a sú rovnaké.

Dôkaz: Na obrázku Obr. 7 sú znázornené výpočty v BP, ako to predpokladá táto lema.

1) Po dosiahnutie K výpočet na c sleduje výpočet a ďalej sa ide po b.
2) Nech to tak nie je, potom existuje premenná x_i, na ktorú sa odkazuje vo výpočte pre b pred dosiahnutím Ka zároveň vo výpočte a po dosiahnutí K. Urobíme vstup c nasledovne, $c =_{b_K} b$ a $c =_{\overline{b_K}} a$, výpočet na tomto vstupe sleduje najprv ten na b až po K, potom sleduje ten pre a (musí to tak dopadnúť, lebo v BP_1 sú všetky vetvy dosiahnuteľné). Čiže výpočet pre c sa odkazoval aspoň dva krát na x_i čo je v spore s tým, že P je read-once. Táto časť lemy hovorí o tom, že premenné z b_K sú po K nedosiahnuteľné nielen pre výpočet na b (čo je z definície read-once), ale aj pre pokračovanie výpočtu na a.
3) Podľa zadefinovania c sa výpočty a a c môžu rozísť len vo vrcholoch s premennou s indexom z $b_K - a_K$, tie sú podľa 2) nedosiahnuteľné pre ich výpočty.



Obr. 7: Schéma výpočtov BP na a a b, ktoré sa stretnú vo vrchole K.

Samotná technika je v podstate pozorovanie intuitívnej vlastnosti, že BP_1 nemá možnosť testovať premennú viac ako raz. Preto, pre zložitejšie funkcie, musí byť dostatočne rozsiahli, aby si mohol v jednotlivých vetvách pamätať hodnoty už otestovaných premenných. Ilustrujeme to na príklade pre $ECLIQUE_n$ (definícia 3.2.3).

Idea je nasledovná. Keď sa v BP_1 P testuje posledný vrchol patriaci do kliky, musí sa zistiť aké sú ostatné vrcholy patriace do kliky. Keďže tie sa už testovali, nemôže sa na ne P odkázať priamo a teda P si to musí pamätať v rozdielnych vetvách výpočtu, čo má za následok veľa vrcholov v P a teda aj veľkú veľkosť. Teraz sa na dôkaz pozrieme detailnejšie.

Lema 4.3.2: (21) Nech P je BP_1 pre jazyk $ECLIQUE_n$, nech a, b sú vstupy, pričom $ECLIQUE_n(a) = 1$ a nech K je spoločný vrchol pre oba výpočty na a aj b. Potom $b_K - a_K = \emptyset$.

Dôkaz: Lema hovorí, že ak aj b obsahuje half-kliku, potom $b_K = a_K$. Nech teda lema neplatí, potom $b_K - a_K \neq \emptyset$. Nech $s \in b_K - a_K$. Urobme vstup c, jeho bit $c_s = \overline{a_s}$ a ostatné bity v c necháme ako má a. Takýto vstup spĺňa predpoklady lemy 4.3.1 časť 3) a teda platí, výpočty na a a c sú rovnaké (oba akceptujú). Čo je ale spor, lebo c určite nie je klika.

Veta 4.3.4: (21)

$$BP_1(ECLIQUE_n) \geq \frac{2^{n/3}}{n/3+1}$$

Dôkaz: Pre jednoduchosť uvažujme n deliteľné 6. Nech P je ľubovoľný BP_1 pre jazyk $ECLIQUE_n$. Nech M je množina vstupov a, pre ktoré $ECLIQUE_n(a) = 1$. Urobíme zobrazenie F z množiny M do vrcholov P. Pre $a \in M$ bude platiť $F(a) = v$, kde v je prvý vrchol taký, že hrany na ktoré sa odkazoval výpočet pre a pred dosiahnutím v pokrývajú aspoň $n/2 - 2$ vrcholov kliky v grafe a. Všimnime si, že ostane ešte 1 alebo 2 nepokrytých vrcholov.

Ďalej nech S je množina všetkých vstupov a, pre ktoré funkcia $ECLIQUE_n$ vracia 1 a pre ľubovoľné dva vstupy z S sa kliky na nich líšia aspoň v 6 vrcholoch. Odhadneme počet prvkov množiny S zdola nasledovne. Rozdelíme si vrcholy grafu do trojíc a urobíme z nich trojuholníky. Číslo $\binom{n/3}{n/6}$ je počet rôznych výberov polovice zo všetkých trojuholníkov. Keď z každého výberu urobíme kompletný graf, bude to half-klika v pôvodnom grafe. Zároveň dva rôzne výbery sa líšia aspoň v 2 trojuholníkoch a teda aspoň v 6 vrcholoch.

$$|S| \geq \binom{n/3}{n/6} \geq \frac{2^{n/3}}{n/3+1}$$

Ak sa nám podarí ukázať, že zobrazenie F je na množine S injektívne, potom sme ukázali požadovanú hranicu pre P. Keďže ten je ľubovoľný, tak aj tvrdenie vety. Pre spor predpokladajme, že to tak nie je.

Nech $a, b \in S$, $a \neq b$ a zároveň $F(a) = F(b) = K$. Podľa lemy 4.3.2 platí $a_K = b_K$, ďalej nech c je vstup taký, že $c =_{a_K} a$ a $c =_{\overline{a_K}} b$. Podľa lemy 4.3.1 časť 1) potom výpočet b je ten istý ako pre c a teda c obsahuje half-kliku. Nech v je vrchol, ktorý je v klike a, ale nie je v klike b (také sú aspoň 3, lebo kliky v S sa líšia v najmenej 6 vrcholoch) a zároveň je pokrytý aspoň jednou hranou z a_K (podľa definície F tam taký bude). Nech u je naopak vrchol z kliky v b, ale nie je v klike a a zároveň u nie je pokrytý žiadnou hranou z $a_K = b_K$ (znova tak taký určite je). Hrany v c určite pokrývajú vrcholy v aj u. Ale keďže c obsahuje iba hrany z a, alebo b určite neexistuje hrana medzi vu. Čo je v spore s tým, že c obsahuje half-kliku. ■

Odhad vo vete 4.3.4 je závislý od parametru funkcie a nie od dĺžky vstupu, ak ako m označíme dĺžku vstupu pre $ECLIQUE_n$, potom dolný odhad pre BP_1 je:

$$BP_1(ECLIQUE_n) \geq 2^{\sqrt{2m}/3 - \log(\sqrt{2m}/3)}$$

Za spomenutie ešte stojí, že Wegenerov jazyk $k - CLIQUE_n$ je NP-úplný problém (čiže nepatrí do P, ak $NP \neq P$). Jazyk $ECLIQUE_n$, ktorý použil Žák, je v P.

Doposiaľ prezentované techniky mali spoločný fakt, že v istom zmysle zisťovali počet vrcholov v BP_1, ktoré musia byť rôzne. Tým sa potom určí dolný odhad pre zložitosť. Toto vyvrcholilo až to vety, prvýkrát dokázanú v (22), tu prezentujeme špeciálny prípad (stále ale dosť všeobecný) vety (23).

Nech I podmnožina množiny premenných X_n. Symbolom $\nu(f, I)$ označme maximálny počet ekvivalentných $(X_n \setminus I)$-podfunkcií funkcie f (definícia 4.1.2) pre rôzne dosadenia konštánt za premenné z I. Pod pojmom čiastočný vstup u definovaný na I rozumieme vstup, kde premenné z I sú konkrétne hodnoty z množiny $\{0,1\}$ a ostatné sú voľné. Ak f je funkcia, symbolom $f|_u$ označíme podfunkciu f, ktorá vznikla dosadeným konštánt z u za premenné, kde je u definovaný, do f. $\nu(f, I)$ sa teda dá formálne zadefinovať ako (S označíme množinu všetkých čiastočných vstupov definovaných na I)

$$\nu(f, I) = \max_{u \in S} |\{v \in S | \, f|_u \equiv f|_v\}|$$

Veta 4.3.5: (23) Nech $f \in B_n$ je funkcia závislá od všetkých svojich premenných a nech $r \leq n$ je prirodzené číslo. Potom ľubovoľný BP_1 počítajúci funkciu f má veľkosť aspoň:

$$\frac{2^{n-r}}{\max\limits_{|I|=n-r} \nu(f, I)}$$

Dôkaz: Nech P je BP_1 realizujúci f. Pre každý vrchol w z grafu P definujeme w^+ ako množinu všetkých premenných, ktoré sa vyskytujú vo w, alebo na ľubovoľnej orientovanej ceste od w k niektorému listu (čiže podprograme, ktorého koreňom je w). Ďalej pre každý vstup x, definujeme zobrazenie $e(x)$ ako hranu uv, kde uv sa nachádza na ceste, ktorú aktivuje výpočet na x na P a platí $|u^+| \geq r + 1$ a $|v^+| \leq r$. Pre každé x taká hrana existuje a je jediná. Existencia je zaručená tým, že f závisí od

všetkých premenných a teda veľkosť grafu P je aspoň n, čiže sa postupným skúšaním všetkých hrán na výpočte pre x dá nájsť hrana s požadovanou vlastnosťou. Jedinečnosť vyplýva z faktu, že výpočet pre x je orientovaná cesta.

Zoberme konkrétny vstup x, ideme odhadnúť počet vstupov y, pre ktoré platí $e(x) = e(y)$. Nech $e(x) = uv$ a nech premenná čítaná v u je x_i. Urobíme množinu J o veľkosti r, aby platilo $u^+ \setminus \{x_i\} \supseteq J \supseteq v^+$, určite taká množina existuje. Ďalej $I = X_n \setminus J$. I nie je prázdna množina, aspoň x_i tam patrí. Pre každý vstup y označme ako y^* čiastočný vstup definovaný na I, kde $y =_I y^*$. Teda ak vstup y aktivuje cestu, ktorá obsahuje hranu uv. Potom všetky premenné testované pred dosiahnutím u (vrátane u) majú rovnakú hodnotu v y aj y^*, preto aj y^* dosiahne hranu uv. Platí to aj naopak, ak y^* dosiahne hranu uv potom aj y.

Ak $e(x) = e(y)$ potom súčasne výpočty y^*, x^* vedú do v. Preto pre každý čiastočný vstup z, definovaný na J, platí $f(y^*, z) = f(x^*, z)$[10], čo vlastne znamená, že tieto dve J-podfunkcie sú ekvivalentné. Takýchto rôznych y^* spĺňajúcich uvedenú vlastnosť je najviac $v(f, I)$. Z toho vyplýva, že vstupov y s vlastnosťou $e(x) = e(y)$ je najviac $v(f, I)2^r$ (každému čiastočnému vstupu y^* môžeme doplniť ešte r premenných, čo dáva 2^r možností). Čiže existuje najviac $v(f, I)2^r$ vstupov y, ktoré majú rovnakú hodnotu $e(y)$.

Vyššie uvedený odhad platí pre každý vstup x a nejakú množinu I o veľkosti $n - r$ (aby sme vedeli číslo $v(f, I)$). Platí, že všetkých vstupov x je 2^n a $v(f, I) \leq \max_{|I|=n-r} v(f, I)$ pre ľubovoľné I. Preto v P musí byť určite aspoň $2^n / \left(2^r \cdot \max_{|I|=n-r} v(f, I)\right)$ hrán (2^n je počet všetkých vstupov, pre každý je definovaná funkcia $e(x)$ a najviac $2^r \cdot \max_{|I|=n-r} v(f, I)$ vstupov má hodnotu $e(x)$ rovnakú). Dostali sme odhad na počet hrán, nás ale zaujíma veľkosť P – počet vrcholov. Keďže v P má každý vnútorný vrchol 2 výstupné hrany musí byť vrcholov aspoň polovica z počtu hrán. Čo nám dalo o ½ horší odhad ako hovorí veta. Odhad sa dá vylepšiť o faktor 2, pomocou istých vlastností grafov. Presný dôkaz je uvedený v (23). ■

Veta 4.3.5 vlastne hovorí, že hľadanie dolného odhadu pre BP_1 sa dá zredukovať na hľadanie horného odhadu pre $v(f, I)$. Savický a Žák (23) jej použitím ďalej ukázali existenciu postupnosti booleovských funkcií $\{f_n\}_{n=1}^{\infty}$, ku ktorej prislúchajúci jazyk je v P (čiže ju považujeme, za explicitne definovanú) a zároveň BP_1 pre funkciu f_n potrebuje veľkosť aspoň $2^{n-3\sqrt{n}}$.

Na read-once BP bolo dokázaných ešte veľa dolných odhadov. Väčšina pre jazyky týkajúce sa klík. Ale podarili sa aj iné jazyky, napríklad Ponzio (24) prezentoval dolný

[10] Keďže čiastočné vstupy y^* a z sú definované na disjunktných množinách, ktorých zjednotením je X_n, $f(y^*, z)$ je už presne určená hodnota funkcie.

odhad $2^{\Omega(\sqrt{n})}$ pre násobenie[11] na BP_1. V (25) je tento odhad zlepšený použitým vety 4.3.5 na $2^{\Omega(n/4)}$. Pre niektoré problémy boli ukázané aj presné odhady zložitosti na BP_1. V (7) je ukázaná zložitosť jazyka SA_n (definícia v kapitole 1.3 príklad 1) ako $BP_1(SA_n) = 2 \cdot 2^n - 1$. Doposiaľ asi najlepším výsledkom je hranica $2^{n-O(\log n)}$ uvedená v (26).

4.4 Odhady pre read *k*-times only branching programy

Pre read-once BP bolo vyvinutých veľa techník pre dolné odhady a ukázať exponenciálne hranice pre niektoré funkcie sa dalo celkom ľahko. Pre BP_k ($k \geq 2$) sú odhady už ťažšie, podarilo sa ale niečo dokázať.

Borodin, Razborov a Smolensky (27) po prvý krát ukazujú netriviálny odhad pre NBP_k, kde k je ľubovoľné. Odhad je založený na vete, ktorá hovorí že každá funkcia f sa dá rozložiť na istý počet podfunkcií, ktoré závisia iba od niektorých premenných.

Veta 4.4.1: (27) Nech $f \in B_n$ je funkcia a k, a sú prirodzené čísla. Nech $T = \left(2NBP_k(f)\right)^{2ka}$, potom f sa dá rozložiť do formy

$$f = \bigvee_{i=1}^{T} \bigwedge_{j=1}^{ka} f_{ij}(X_{ij})$$

Kde f_{ij} závisí iba od premenných z $X_{ij} \subseteq X_n$, $|X_{ij}| = \lfloor n/a \rfloor$ a zároveň pre každé i ($1 \leq i \leq T$), každá premenná patrí do najviac k množín z $\{X_{i,1}, \dots, X_{i,ka}\}$.

Pomocou tejto vety dosiahli pre niektoré funkcie f odhady $NBP_k(f) \geq 2^{\Omega(n/4^k k^3)}$, hranica sa zhoršuje podľa toho ako rastie k, pre $k \geq \log n$ to už nie je exponenciálny odhad.

Okol'nishnikova v svojej práci (28) ukázala aké dôležité je získať dobré odhady pre NBP_k, bez nich sa totiž nedajú získať dobré odhady pre všeobecný prípad NBP. Idea je, že ak sa v NBP veľakrát číta niektorá premenná, potom tento NBP nemôže byť malý.

Veta 4.4.2: (28) Nech $f \in B_n$, C je konštanta ($0 < C < 1$), $\psi(n)$ je rastúca funkcia. Ďalej nech X_0 je podmnožina premenných X_n taká, že $|X_0| = \lfloor Cn \rfloor$. Ak pre všetky X_0 existuje taká substitúcia konštánt za premenné v X_0, že funkcia g vzniknutá touto substitúciou z funkcie f (g je podfunkcia f) má zložitosť $NBP_{k(n)}(g) \geq n\psi(n)$, potom $NBP(f) \geq \min\{Cnk(n), n\psi(n)\}$.

[11] Pod násobením si môžeme predstaviť funkciu $MUL(a, b)$, kde výsledok je 1, ak stredný bit v $a \cdot b$ je 1, inak je výsledkom 0.

Dôkaz: Nech P je NBP počítajúci f ukážeme, že pre jeho zložitosť platí odhad z vety. Symbolom Y označíme množinu premenných, na ktoré sa P odkazuje najmenej $k(n)$ krát. Môžu nastať dva prípady.

$|Y| > \lfloor Cn \rfloor$, teda určite aspoň $\lceil Cn \rceil$ premenných sa v P vyskytuje viac ako $k(n)$, z čoho vidieť, že P nemôže mať menej ako $\lceil Cn \rceil k(n) \geq Cnk(n) \geq \min\{Cnk(n), n\psi(n)\}$ vrcholov a teda tvrdenie vety platí.

$|Y| \leq \lfloor Cn \rfloor$, ak $|Y| < \lfloor Cn \rfloor$, potom doplníme množinu Y premennými, aby sme dostali množinu X_0 o veľkosti $\lfloor Cn \rfloor$. Podľa predpokladov vo vete existuje substitúcia konštánt do f za premenné v X_0 taká, že výsledná funkcia g má zložitosť aspoň $NPB(g) \geq n\psi(n)$. Ľahko vykonáme túto substitúciu aj do nášho programu P, dostaneme program R, ktorého zložitosť nie je väčšia ako zložitosť P (substitúciou len odstraňujeme vrcholy a žiadne nepridávame). R je zároveň read $k(n)$-times only NPB, nakoľko všetky premenné s viac výskytmi sme odstránili v substitúcií. R má určite aspoň $n\psi(n)$ vrcholov, čiže P ich nemôže mať menej. Tým sme dokázali vetu. ■

V dôkaze sa nikde nevyskytuje vlastnosť, že BP by mal byť nedeterministický, preto veta platí aj pre deterministický prípad. Touto technikou sa podarilo získať netriviálny odhad (28) na všeobecných NPB pre charakteristickú funkciu f Reed-Mullerovho kódu $NPB(f) \geq \Omega(n \log n / \log\log n)$ (kde n je dĺžka vstupu).

Definícia 4.4.1: Ak X je zložitostná trieda, potom $co - X$ je trieda problémov, ktorých komplement je v X.

Jukna (29) ukázal ďalšiu hranicu na NPB_k, konkrétne použil funkciu

$$f_{n,d}(x_1, \dots, x_n) = \bigvee_{i=1}^{m} \left(1 \oplus \left(\oplus_{j=1}^{n} a_{ij} x_j\right)\right)$$

kde $A = \{a_{ij}\}$ je booleovská matica $m \times n$ ($m \leq d \log(n+1)$) taká, že každých $2d$ stĺpcov je lineárne nezávislých.

Pre $f_{n,d}$ sa mu podarilo dokázať $NPB_k(f_{n,d}) = 2^{\Omega(\sqrt{n}/k^{2k})}$. Ďalej si všimol, že funkcia $\overline{f_{n,d}}$ sa dá realizovať na NPB_1 s polynomiálnou veľkosťou, zatiaľ čo $f_{n,d}$ sa nedá definovať na $NPB_k(P)$[12] ak $k \leq k_0 = (1/2 - \varepsilon) \ln n / \ln\ln n$. Čo má za následok

$$co - NBP_1(P) \setminus NBP_{k_0}(P) \neq \emptyset$$

Jukna tým ukázal, že $NBP_k(P) \neq co - NBP_k(P)$, pre $k \leq k_0$. Z Immermanovej (30) a Szelepcsenyho vety vieme, že nedeterministický priestor je uzavretý na komplement, čo znamená, že $NBP(P) = co - NBP(P)$. Jukna vlastne dokázal, že rovnosť určite potrebuje aspoň logaritmický počet čítania jedného bitu vstupu pre nedeterministický logaritmický priestor.

[12] Trieda nedeterministických read k-time only BP s polynomiálnou veľkosťou od dĺžky vstupu.

4.5 Odhady pre read $(1, +k)$-branching programy

V tento kapitole uvedieme jeden výsledok pre dolné odhady na $(1, +k) - BP$, ukáže sa, že pre funkcie istých parametrov, nemôže existovať malý program. Túto vlastnosť použili autori v (31), odtiaľ čerpáme vetu a jej dôkaz.

Zavedieme niektoré označenia. Nech a je čiastočný vstup, potom $S(a) \subseteq X_n$ je množina premenných, ktoré sú v a zafixované (a je na nich definovaný). Ďalej nech a, b sú čiastočné vstupy, potom $D(a, b)$ označíme premenné z $S(a) \cap S(b)$, ktoré majú v a rôznu hodnotu od b. Pojmom $minterm$ ($maxterm$) funkcie f označíme čiastočný vstup a, ak platí $f|_a \equiv 1$ (resp. $f|_a \equiv 0$) a zároveň a je minimálny v zmysle, že ak v a uvoľníme jednu zafixovanú premennú, už nebude po substitúcií platiť podmienka ekvivalencie s konštantou 1 (0). Funkcia $comp(a)$ bude definovaná na čiastočných vstupoch a vracia prvý vrchol BP, v ktorom sa testuje hodnota nedefinovaná v a a dostane sa doňho výpočet na a.

Definícia 4.5.1: (31) Funkcia f je $d - rare$, ak pre ľubovoľné dva rôzne vstupy a, b také, že $f(a) = f(b) = 1$, platí $|D(a, b)| \geq d$. Ďalej f je $m - dense$, ak pre každý $maxterm$ a funkcie f platí, že počet zafixovaných premenných v a (dĺžka a) je aspoň m.

Ako hovorí názov $d - rare$ funkcia f má vstupy, na ktorých f dáva 1, od seba vo vzdialenosti aspoň d. Vzdialenosť meria vlastne funkcia D. Pre $m - dense$ funkciu, musíme poznať určite aspoň m hodnôt premenných, aby sme mohli z istotou určiť hodnotu funkcie ako 0.

Definícia 4.5.2: (31) Nech a, b sú čiastočné vstupy ($S(a) = S(b)$) a nech P je BP. a, b nazveme „zabúdacím“ párom, ak v P existuje vrchol v, patriaci čiastočným výpočtom $comp(a)$ aj $comp(b)$ a oba výpočty sa odkazovali na premenné z $D(a, b)$ aspoň raz.

Zabúdacie preto, lebo vo v sa zabudli všetky rozdiely medzi a a b. Aby sa znova zistili, musia sa príslušné premenné prečítať znova. Nasledujúca lema hovorí o podmienkach existencie „zabúdacích“ párov v BP.

Lema 4.5.1: (31) Nech P je BP, kde sa každý výpočet odkazuje na aspoň m rôznych premenných. Nech s je prirodzené číslo a platí $1 \leq s \leq \frac{m}{2\log_2|P|+1}$. Potom existujú, po dvoch disjunktné, množiny $I_j \subseteq X_n$ pre $1 \leq j \leq s$ a čiastočné vstupy $a_j \neq b_j$, pre ktoré platí $S(a_j) = S(b_j) = I_j$ také, že pre všetky $1 \leq j \leq s$ platí:

1) $|I_j| \leq 2\log_2|P| + 1$
2) Vstupy $(a_1, \dots, a_j)$ a $(a_1, \dots, a_{j-1}, b_j)$ tvoria „zabúdajúci“ pár.

Veta 4.5.1: (31) Nech d, m, k sú prirodzené čísla z intervalu $(1 \dots n)$. Potom pre ľubovoľný $(1, +k) - BP$ počítajúci $d - rare$ a $m - dense$ funkciu f platí:

$$(1, +k) - BP\ (f) \geq 2^{(\min\{d,\, m/(k+1)\}-1)/2}$$

Dôkaz: Sporom predpokladajme, že existuje $(1, +k) - BP\ P$ počítajúci funkciu f s menšou veľkosťou ako hovorí veta. Ďalej môžeme predpokladať, že $d \geq 2$, inak by sa odhad stal triviálnym. Z toho priamo vyplýva, že existuje vstup na ktorom dáva f hodnotu 0. Ak by neexistoval, tak $d = 1$. Podľa definície máme, že dĺžka každého $maxterm$-u je aspoň m. Pre každý $minterm$ musí platiť, že má dĺžku aspoň n. Keby to tak nebolo, potom by existoval $minterm$ s dĺžkou menej ako n, čiže by existovali aspoň dva vstupy, ktoré sa líšia v jednom bite a f má na nich hodnotu 1, čo je v spore s tým, že $d \geq 2$ a f je $d - rare$. Platí $n \geq m$, teda pre zistenie hodnoty f musíme pozrieť aspoň m premenných, z čoho vyplýva, že aj náš P musí na každej výpočtovej vetve otestovať aspoň m rôznych premenných. Veľkosť P je menšia ako $2^{(\min\{d,\, m/(k+1)\}-1)/2}$, teda

$$\log_2|P| < \log_2 2^{\left(\min\left\{d,\ \frac{m}{(k+1)}\right\}-1\right)/2} < \frac{1}{2}\left(\frac{m}{k+1} - 1\right)$$

$$k + 1 < \frac{m}{2\log_2|P| + 1}$$

Čiže P spĺňa predpoklady lemy 4.5.1, pre $s = k + 1$. Čiže existujú množiny I_j a vstupy $a_j,\ b_j$ $(1 \leq j \leq k + 1)$ s vlastnosťami 1) a 2) uvedenými v leme. Z 1) vyplýva, že pre každú I_j platí $\left|I_j\right| \leq 2\log_2|P| + 1 < \min\{d,\ m/(k+1)\}$. Z toho vyplýva, že čiastočný vstup $(a_1, \dots, a_{k+1})$ definuje ostro menej ako m premenných $((k+1) \cdot \left|I_j\right| < m)$. Pretože f je $m - dense$ musí sa dať $(a_1, \dots, a_{k+1})$ rozšíriť (dodefinovaním voľných premenných) na vstup a, pre ktorý platí $f(a) = 1$. Keby to tak nebolo, potom $(a_1, \dots, a_{k+1})$ by pre všetky rozšírenia dával v f hodnotu 0 a teda by existoval $maxterm$ kratší ako m (spor).

P je $(1, +k) - BP$ preto môže iba k premenných čítať viac ako raz, z toho hneď vyplýva, že musí existovať j také, že premenné z množiny I_j sa testujú najviac raz na ceste výpočtu $comp(a)$. Z vlastnosti 2) lemy vieme, že vstupy $\left(a_1, \dots, a_j\right)$ a $\left(a_1, \dots, a_{j-1}, b_j\right)$ tvoria „zabúdací“ pár. Nech v je zodpovedajúci vrchol (definícia 4.5.2). Potom $v \in comp(a)$ a zároveň premenné z $D\left(a_j, b_j\right)$ (ostatné premenné sú v čiastočných vstupoch rovnaké) boli testované aspoň raz na $comp(a)$ pred dosiahnutím v. Platí $D\left(a_j, b_j\right) \subseteq I_j$, lebo a_j aj b_j sú definované na I_j. Urobíme vstup b, ten vznikne z a nahradením premenných a_j za b_j. $comp(b)$, určite dosiahne vrchol v (tak je definovaný „zabúdací“ pár) a ďalej sleduje výpočet $comp(a)$, čiže b dosiahne rovnaký list ako a, preto $f(b) = 1$. $D(a, b) = D\left(a_j, b_j\right) \subseteq I_j$, preto aj $|D(a, b)| \leq \left|I_j\right| < d$. Posledná nerovnosť vyplýva z faktu $\left|I_j\right| < \min\{d,\ m/(k+1)\}$. Čo je ale v spore s vlastnosťou $d - rare$ funkcie f. ■

Pomocou tejto vety sa hľadanie dolných odhadov dá premeniť na hľadanie vlastností $rare$ a $dense$ funkcií. Autori ukázali, že tieto vlastnosti spĺňajú charakteristické funkcie lineárnych kódov. Konkrétne pre niektoré Reed-Mullerove kódy sa podaril ukázať odhad $2^{\Omega\left(\sqrt{n/k+1}\right)}$ pre veľkosť $(1,+k)-BP$.

4.6 Ďalšie dolné odhady

Existuje ešte veľa odhadov na zložitosť, ktoré sme nespomenuli. Vytvorili sa aj viaceré modely so značnými ohraničeniami. Menovite hlavne OBDD (Ordered binary decision diagrams), o nich viac v (6) a (32). Ďalej balanced BP spomenuté v (33) a mnoho iných. Autori dúfali, že dokazovaným odhadov v ohraničených prípadoch budú schopný vyvinúť techniku pre všeobecný prípad, žiaľ nestalo sa tak. Na záver uvedieme jeden výsledok ktorý sa predsa podaril a to odhady pre symetrické funkcie (definícia 4.2.2). Ukázali ho autori Pudlák, Aitai, Babai, Hajnal, Kolmós, Rödl, Szemerédi a Turán vo svojej práci (34).

Veta 4.6.1: (34) Nech f je skoro (až na exponenciálne malú časť) ľubovoľná symetrická funkcia. Nech P je leveled-BP počítajúci f so šírkou $O(\log n)^c$, kde c je ľubovoľná konštanta. Potom veľkosť P je najmenej $n \log n/C \log\log n$, pre nejakú kladnú konštantu C.

Veta platí napríklad pre funkciu f definovanú nasledovne. Nech p je prvočíslo také, že $n^{1/4} < p < n^{1/3}$. Potom $f(x) = 1$ práve vtedy, keď počet 1 v x je kvadratické rezíduum modulo p (existuje a také, že $a^2 \equiv x\ (mod\ p)$).

Autori prácu zakončili hypotézou, že použitím rovnakých metód opakovane sa im podarí zlepšiť odhad až na $\Omega(n \log n)$. Zároveň dúfali v možnosť odstránenia ohraničenia šírky s tým, že im stále ostane netriviálna hranica. Toto sa im podarilo a v (35), kde naozaj ukázali hranicu $\Omega(n \log n)$ pre symetrické funkcie na BP s ohraničenou šírkou a $\Omega(n \log n/\log\log n)$ pre BP bez ohraničení. Keďže každá symetrická funkcia sa dá realizovať BP s veľkosťou $O(n^2/\log n)$ (10) ešte stále je medzera medzi najlepším odhadom zhora a zdola.

Kapitola 5

Natural Proofs

V predchádzajúcej kapitole sme uviedli, že na BP sa nedarí získať vyššie dolné odhady. Obdobne je to aj v prípade s booleovskými obvodmi. Pretrváva problém získať superpolynomiálne dolné odhady pre explicitne definované funkcie. Iné problémy na tom nie sú o nič lepšie. V teórií zložitosti je veľa podstatných otázok, na ktoré zatiaľ nepoznáme odpoveď.

Príkladom je najslávnejší otvorený problém $P =? NP$. Problém, známy už niekoľko desaťročí, ktorým sa zaoberalo množstvo matematikov a informatikov. Žiaľ problém je stále nevyriešený. Samotná jeho definícia je ľahko formulovateľná. Formálne sme triedy P a NP zadefinovali v kapitole 2.4. Neformálne sa často o nich rozpráva ako o triedach „easy to find“ (problémy, ku ktorým vieme ľahko nájsť riešenia) a „easy to verify“ (ak poznáme riešenie, potom vieme ľahko overiť, či je správne). Po mnohých neúspešných snahách dokázať rovnosť alebo ich odseparovať, prišli vedci s myšlienkou, že možno sa ich vzťah ani nedá dokázať v rámci teórie, ktorú používame (problém je nezávislý od štandardného axiomatického systému teórie množín).

Vývoj pokračoval ďalej až do okamihu, keď sa objavili úvahy o bariérach. Každý dôkaz o P vs. NP musí bariéry prekonať. Bariéry sú:

1) relativizácia – hovorí, že techniky ako diagonalizácia nie sú dosť silné na rozhodnutie P vs. NP
2) algebraizácia – algebrická relativizácia
3) natural proofs

Posledne menovaná súvisí s BP a dolnými odhadmi zložitosti, preto sa jej budeme ďalej bližšie venovať. Viac o probléme nezávislosti P vs. NP sa dá nájsť vo vynikajúcej práci S. Aaronsona (36).

Natural proofs je koncept po prvý krát prezentovaný v práci Razborova a Rudicha (3). Ukázali, že všetky dolné odhady na neuniformných modeloch pre booleovské funkcie sú v istom zmysle urobené jedinou prirodzenou technikou. Neformálne postupuje nasledovne. Chceme pre funkciu $f_n \in B_n$ ukázať, že patrí do istej triedy S (napríklad to môže byť PBP).

1) Zadefinujeme nejakú vlastnosť booleovských funkcií C_n.
2) Ukážeme, že pre všetky funkcie g_n, ktoré majú vlastnosť C_n nepatria do S.
3) Ukážeme, že aj f_n má C_n a dôkaz je u konca, lebo určite f_n nie je v S.

Formálne, vlastnosť C_n je podmnožina B_n. Hovoríme, že funkcie, ktoré patria do C_n túto vlastnosť majú a naopak funkcie, ktoré je nemajú nepatria do C_n.

5.1 Prirodzené vlastnosti

Vlastnosť $C_n \subseteq B_n$ je Γ-prirodzená, ak existuje $C_n^* \subseteq C_n$ (veľmi často $C_n^* = C_n$) a sú splnené vlastnosti:

1) konštruovateľnosť – rozhodnúť, či funkcia f_n má vlastnosť C_n^* ($f_n \in C_n^*$) sa dá v zložitostnej triede Γ (funkcie reprezentujeme pomocou pravdivostných tabuliek, tie majú veľkosť 2^n). Napríklad ak $\Gamma = \mathrm{P}$, potom rozhodnúť, či f_n patrí do C_n^* sa dá v polynomiálnom čase od 2^n.
2) veľkosť – musí platiť $|C_n^*| \geq 2^{-O(n)} \cdot |B_n| = 2^{-O(n)} \cdot 2^{2^n}$.
3) užitočnosť – vlastnosť C_n je užitočná voči zložitostnej triede S, ak pre každú postupnosť funkcií $f_1, \ldots, f_n, \ldots$, kde udalosť $f_n \in C_n$ nastáva nekonečne často, platí $\{f_n\}_{n=1}^{\infty} \notin S$.

Ukázalo sa, že skoro všetky používané vlastnosti sú konštruovateľné a majú požadovanú veľkosť. Užitočnosť vlastnosti sa vždy berie vzhľadom na konkrétnu zložitostnú triedu S. Často sa za triedu S berie $P/Poly$ (čo je aj trieda neuniformných booleovských obvodov polynomiálnej veľkosti (1)). Potom je vlastnosť C_n užitočná, ak pre každú postupnosť funkcií $f_1, \ldots, f_n, \ldots$ ($f_n \in C_n$ nastáva nekonečne často), platí $\{f_n\}$ nemá obvod s polynomiálnou veľkosťou.

Otázka je, ktorý argument potom nie je prirodzený. Napríklad by takým mohlo byť jednoduché spočítavanie veľkosti množín. Vlastnosť C_n zadefinujeme ako f_n má C_n, ak f_n nemá polynomiálny BP. Množina C_n je určite veľká (podľa vety 4.0.1), ale to nám nepomôže pri konštruovaní C_n^*. Táto vlastnosť je z nášho pohľadu neprirodzená. To nám neprekáža, lebo pomocou spočítavania ešte nebol urobený žiadny dolný odhad pre explicitne definovanú funkciu.

Dôkazy, ktoré používajú prirodzené vlastnosti sa nazývajú natural proofs (prirodzené dôkazy).

5.2 Limity natural proofs

Najzaujímavejšie na práci (3) je, že autori prišli s dôkazom, podľa ktorého je sila prirodzených dôkazov inherentne obmedzená. Predstavme si, že chceme pre f ukázať $f \notin P/Poly$. Potom každý prirodzený dôkaz musí odlíšiť f od ľubovoľnej pseudonáhodnej funkcie z $P/Poly$, algoritmus použitý v dôkaze, ale musí fungovať pre každú funkciu. Ak za f zvolíme náhodnú funkciu (ktorá určite nepatrí do $P/Poly$), algoritmus použitý na rozlíšenie f od funkcií z $P/Poly$ je vlastne algoritmus na prelomenie pseudonáhodného generátora čísel z triedy $P/Poly$.

Definícia 5.2.1: (3) Pseudonáhodný generátor čísel je booleovská funkcia na n premenných $G_n: \{0,1\}^n \longrightarrow \{0,1\}^{2n}$. Silou pseudonáhodného generátora $H(G_n)$ označujeme najmenšie prirodzené číslo s také, že existuje booleovský obvod C s veľkosťou s, pre ktorý platí

$$\left|\mathbf{P}\left[C\left(G_n(x)\right)=1\right]-\mathbf{P}[C(y)=1]\right| \geq \frac{1}{s}$$

kde x je náhodný vstup z $\{0,1\}^n$ a y je náhodný vstup z $\{0,1\}^{2n}$.

Neformálne, sila je najmenšie číslo s, pre ktoré existuje booleovský obvod C s veľkosťou najviac s. Zároveň pravdepodobnosť toho, že C rozlíši medzi pseudonáhodným a náhodným reťazcom je aspoň $1/s$.

Veta 5.2.1: (3) Predpokladajme, že existuje dôkaz dolného odhadu zložitosti, ktorý používa $P/Poly$-prirodzenú vlastnosť užitočnú voči triede $P/Poly$. Potom pre každý generátor pseudonáhodných čísel $G_k: \{0,1\}^k \longrightarrow \{0,1\}^{2k}$, ktorý má polynomiálnu časovú zložitosť, platí $H(G_k) \leq 2^{k^{O(1)}}$.

Dôkaz: Nech C_n je $P/Poly$-prirodzená vlastnosť použitá v dôkaze z predpokladu vety. Potom existuje $C_n^* \subseteq C_n$, ktorá spĺňa podmienky konštruovateľnosti a má požadovanú veľkosť. Pre jednoduchosť budeme uvažovať $C_n^* = C_n$. Nech $c > 0$ je ľubovoľná konštanta a nech $n = \lceil k^c \rceil$. Pomocou generátora G_k urobíme generátor pseudonáhodných funkcií $F: \{0,1\}^k \longrightarrow B_n$ nasledovne.
Označme ako G_0, G_1 booleovské funkcie $G_0, G_1: \{0,1\}^k \longrightarrow \{0,1\}^k$. $G_0(a)$ bude prvých k bitov výstupu $G_k(a)$, podobne $G_1(a)$ bude posledných k bitov $G_k(a)$. Pre reťazec $y \in \{0,1\}^n$ definujeme $G_y^*: \{0,1\}^k \longrightarrow \{0,1\}^k$.

$G_y^* = G_{y_n} \circ G_{y_{n-1}} \circ \ldots \circ G_{y_1}$(kompozícia funkcií G_0, G_1 určená pomocou reťazca y).

Pre $x \in \{0,1\}^k$ definujeme funkciu $F(x)$ na vstupe $y \in \{0,1\}^n$ ako $F(x)(y)$ je rovné prvému bitu výstupu $G_y^*(x)$. Ľahko vidieť, že vyhodnotenie $F(x)(y)$ sa realizuje s polynomiálnou časovou zložitosťou $F(x) \in P/Poly$, keďže G_k má polynomiálnu časovú zložitosť. Pre ľubovoľné, ale fixné $x \in \{0,1\}^k$ je teda funkcia $F(x) \in B_n$ realizovateľná obvodom s polynomiálnou veľkosťou.

Podľa definície užitočnosti voči triede $P/Poly$ určite $F(x) \notin C_n$ (pre dostatočne dlhý vstup x). Inak by $F(x) \notin P/Poly$. Testovanie $F(x) \in C_n$, podľa predpokladu konštruovateľnosti vlastnosti C_n, má polynomiálnu časovú zložitosť od dĺžky pravdivostnej tabuľky $F(x)$, čiže $2^{O(n)} = 2^{O(k^c)}$. Náhodná funkcia $f \in B_n$ s vysokou pravdepodobnosťou nepatrí do triedy $P/Poly$. Preto C_n je vlastne štatistický test, ktorý dokáže rozlíšiť medzi náhodnou a pseudonáhodnou funkciou v čase $2^{O(k^c)}$, naviac platí tvrdenie

$$\left|\mathbf{P}[C_n(f)=1]-\mathbf{P}\left[C_n\left(F(x)\right)=1\right]\right| \geq 2^{-O(n)} = 2^{-O(k^c)}$$

Algoritmus na rozhodovanie $g \in C_n$ (pre ľubovoľnú funkciu $g \in B_n$) je po úpravách aj algoritmom pre rozlíšenie pseudonáhodnej postupnosti generovanej G_k od náhodnej postupnosti. Ďalej sa pomocou štatistických vlastností dokáže tvrdenie vety. Detailný dôkaz je v (3). ■

Problém je, že v $P/Poly$ sa nachádzajú aj veľmi netriviálne pseudonáhodné generátory, ktorých rozbitie musí vyriešiť problémy ako faktorizácia a diskrétny logaritmus. Ak by existoval prirodzený dôkaz používajúci $P/Poly$-prirodzenú vlastnosť užitočnú voči triede $P/Poly$, potom by existoval algoritmus pre riešenie problému faktorizácie s časovou zložitosťou $2^{O(n^c)}$, pre ľubovoľné $c > 0$. Taký algoritmus by bol o dosť lepší ako techniky, ktoré používame v súčasnosti. To je dôvod, prečo sa nedarí pomocou prirodzených dôkazov ukázať, že nejaká funkcia nie je v $P/Poly$ (nemá polynomiálny obvod). Podobne to funguje pre každú zložitostnú triedu S. Na ukázanie toho, že $f \notin S$ by sme museli byť schopní prelomiť akýkoľvek generátor pseudonáhodných čísel z S. Pre obmedzené triedy S sa to môže podariť, ale ak S obsahuje širokú škálu funkcií, už sa objavujú problémy.

Záver

V práci sme predstavili výpočtový model branching programov, okrajovo sme sa zaujímali aj o iné neuniformné modely. V druhej kapitole sme ukázali niekoľko vzťahov platiacich vo svete neuniformných modelov a rozdiely oproti svetu uniformných modelov.

Pokúsili sme sa načrtnúť, prečo sú dôležité dolné odhady na BP a aké dolné odhady zložitosti sa podarilo dokázať. Najznámejšie techniky pre odhady zložitosti sme prezentovali. Existuje mnoho ďalších, na ktoré už nezostalo miesto.

Ako vyplýva z tretej kapitoly, techniky na dolné odhady nie sú také dobré, ako by sme potrebovali. Vieme, že väčšina funkcií je veľmi zložitých, ale keď príde na konkrétne (explicitne definované) funkcie, metódy zlyhávajú. Darí sa iba na veľmi špeciálnych postupnostiach funkcií.

V poslednej kapitole sme predstavili koncept natural proofs (prirodzené dôkazy), čo je hlavný dôvod, prečo sa nedarí získať dolné odhady zložitosti. Skoro všetky dôkazy sa držia schémy natural proofs. Ak pomocou nich ukážeme, že funkcia je ťažká, potom sme zároveň aj vyriešili nejaký ťažký problém. Napriek tomuto faktu existuje progres v odhadoch zložitosti, zatiaľ čo niektoré problémy stále odolávajú.

Autori používajú rozdielne definície pojmov a implicitné predpoklady viet. V práci sme podávali vety a dôkazy s jednotnými definíciami. Zároveň explicitne uvádzame všetky predpoklady k vetám.

Po komunikácií s odborníkmi z oblasti branching programov vznikal, paralelne s tvorbou práce, archív článkov venujúcich sa problematike neuniformných výpočtových modelov (najmä branching programov) a dolných odhadov zložitosti. Vytvorila sa aj verzia práce, kde sú priame hypertextové odkazy na tieto články.

Literatúra

1. **Wegener Ingo.** The Complexity of Boolean Functions. Stuttgart : John Wiley and Sons Ltd, and B. G. Teubner, 1987.
2. **Barrington David.** Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC1. *Proceedings of the eighteenth annual ACM symposium on Theory of computing.* 1986.
3. **Razborov Alexander, Rudich Steven.** Natural proofs. *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing.* 1994, s. 204 - 213.
4. **Lee C. Y.** Representation of Switching Circuits by Binary-Decision Programs. *Bell Systems Technical Journal.* 1959, 38, s. 985–999.
5. **Masek William.** A fast algorithm for the string editing problem and decision graph complexity. 1976.
6. **Wegener Ingo.** Branching Programs and Binary Decision Diagrams - Theory and Application. s.l. : Society for Industrial and Applied Mathematic, 2000.
7. **Bollig Beate.** The Optimal Read-Once Branching Program Complexity for the Direct Storage Access Function. *Information Processing Letters.* 2007, Zv. 4, 106.
8. **Pippenger Nicholas.** On simultaneous resource bounds. *Proceedings of the 20th Annual Symposium on Foundations of Computer Science.* 1979, s. 307-311.
9. **Cobham Alan.** The Recognition Problem for the Set of Perfect Squares. *Proceedings of the 7th Annual Symposium on Switching and Automata Theory.* 1966.
10. **Razborov Alexander.** Lower Bounds for Deterministic and Nondeterministic Branching programs. *Lecture Notes In Computer Science.* 1991, s. 47 - 60.
11. **Borodin Allan.** On Relating Time and Space to Size and Depth. *SIAM J. Comput.* 1977, Zv. 4, 6.
12. **Beame Paul, Saks Michael, Thathachar Jayram.** Time-Space Tradeoffs for Branching Programs. *Electronic Colloquium on Computational Complexity, Report No. 53.* 1998.
13. **Wegener Ingo.** On the Complexity of Branching Programs and Decision Trees for Clique Functions. *Journal of the Association for Computing Machinery.* 32, 1988, Zv. 2.

14. **Thathachar Jayram.** On Separating the Read-k-Times Branching Program Hierarchy. *Proceedings of the thirtieth annual ACM symposium on Theory of computing.* 1998, s. 653 - 662.
15. **Bollig Beate.** Complexity theoretical results on nondeterministic graph-driven read-once branching programs. *Lecture Notes in Computer Science.* 2003, Zv. 2607.
16. **Savický Petr, Žák Stanislav.** A Hierarchy for (1, +k)-Branching Programs with Respect of k. *Lecture Notes In Computer Science.* 1997, Zv. 1295.
17. **Nechiporuk E. I.** On a Boolean function. *Soviet Math. Dokl. 7 (4).* 1966, s. 999-1000.
18. **Schürfeld Ute.** New lower bounds on the formula size of Boolean functions. *Acta Informatica.* 1983, Zv. 19, 2, s. 183-194.
19. **Borodin Allan, Dolev Danny, Fich Faith, Paul Wolfgang.** Bounds for Width Two Branching Programs. *Proceedings of the fifteenth annual ACM symposium on Theory of computing.* 1983, s. 87 - 93.
20. **Yao Andrew.** Lower Bounds by Probabilistic Arguments. *Foundations of Computer Science.* 1983.
21. **Žák Stanislav.** An Exponential Lower Bound for One-Time-Only Branching Programs. *Lecture Notes In Computer Science.* 1984, 176, s. 562 - 566.
22. **Simon Janos, Szegedy Mario.** A New Lower Bound Theorem for Read Only Once Branching Programs and Its Applications. *Electronic Colloquium on Computational Complexity.* 1992.
23. **Savický Petr, Žák Stanislav.** A large lower bound for 1-branching programs. *Electronic Colloquium on Computational Complexity.* 1996.
24. **Ponzio Stephan.** A Lower Bound for Integer Multiplication with Read-Once Branching Programs. *SIAM Journal on Computing.* 1998, Zv. 28, 3.
25. **Bollig Beate, Woelfel Philipp.** A read-once branching program lower bound of Ω(2^n/4) for integer multiplication using universal hashing. *Proceedings of the thirty-third annual ACM symposium on Theory of computing.* 2001.
26. **Andreev Alexander, Baskakov Juri, Clementi Andrea, Rolim José.** Small Pseudo-Random Sets Yield Hard Functions: New Tight Explict Lower Bounds for Branching Programs. *Lecture Notes In Computer Science.* 1999, 1644.
27. **Borodin Allan, Razborov Alexander, Smolensky R.** On lower bounds for read-k-times branching programs. *Computational Complexity.* 1993, Zv. 3, 1.
28. **Okol'nishnikova Elizaveta.** On one lower bound for branching programs. *ECCC Report TR02-020.* 2002.

29. **Jukna Stasys.** A Note on Read-k Times Branching Programs. *RAIRO Theoretical Informatics and Applications.* 1995, Zv. 29, 1.

30. **Immerman Neil.** Nondeterministic space is closed under complementation. *SIAM Journal on Computing.* 1988, Zv. 17, 5.

31. **Jukna Stasys, Razborov Alexander.** Neither Reading Few Bits Twice nor Reading Illegaly Helps Much. *Discrete Applied Mathematics.* 1998, Zv. 85, 3.

32. **Bollig Beate, Sauerhoff Martin, Sieling Detlef, Wegener Ingo.** On The Power Of Different Types Of Restricted Branching Programs. *Electronic Colloquium on Computational Complexity.* 1994.

33. **Jukna Stasys, Žák Stanislav.** On uncertainty versus size in branching programs. *Theoretical Computer Science.* 2003, Zv. 290, 3.

34. **Ajtai Miklós, Babai László, Hajnal Péter, Rödl Vojtech, Komlós János, Pudlák Pavel.** Two Lower Bounds for Branching Programs. 1986.

35. **Babai László, Pudlák Pavel, Rödl Vojtech, Szemeredi E.** Lower bounds to the complexity of symmetric Boolean functions. *Theoretical Computer Science.* 1990, Zv. 74, 3.

36. **Aaronson Scott, Fortnow Lance.** Is P versus NP formally independent? 2003.