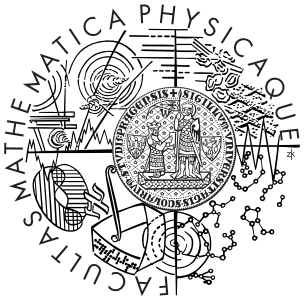
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCA



Štefan Gurský

Časová zložitosť minimalizácie Booleovských funkcí

Katedra teoretické informatiky a matematické logiky

Vedúci diplomovej práce: doc. RNDr. Ondřej Čepek, Ph.D.
Študijný program: Informatika, Teoretická informatika

2010

Rád by som poďakoval vedúcemu práce doc. RNDr. Ondřejovi Čepkovi, Ph.D. za podporu a za to, že organizoval pravidelné stretnutia venované tematike Booleovských funkcií.

Vyhlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa 03.08.2010. Štefan Gurský

Obsah

1 Úvod **6**
- 1.1 Booleovské funkcie 7
- 1.2 Logické spojky 8
- 1.3 Vlastnosti niektorých spojok 9

2 Normálne formy **11**
- 2.1 Základné definície 11
- 2.2 Podobnosti a vzťahy medzi DNF a CNF 13
- 2.3 Hľadanie všetkých primárnych implikantov 16

3 Triedy zložitosti **19**
- 3.1 Ľahké problémy 19
- 3.2 Ťažšie problémy 21
- 3.3 Prevody a úplnosť 22
- 3.4 Orákulá a hierarchia 23

4 Minimalizácia **26**
- 4.1 Dôvody a miery 26
- 4.2 Veľkosti niektorých funkcií 28
- 4.3 Ekvivalencia formulí 29

5 Minimalizácia DNF – Umansov dôkaz **31**
- 5.1 Σ_2 a Shortest Implicant Core 31
- 5.2 Minimalizácia výskytov literálov 35
- 5.3 Minimalizácia počtu termov 37

6 Matched formuly **42**
- 6.1 Základné pojmy a vlastnosti 42
- 6.2 Zložitosti u Matched 43

6.3 Otvorené problémy 49

Literatúra **50**

Názov práce: Časová složitost minimalizace Booleovských funkcí.
Autor: Štefan Gurský
Katedra: Katedra teoretické informatiky a matematické logiky
Vedúci diplomovej práce: doc. RNDr. Ondřej Čepek, Ph.D.
e-mail vedúceho: Ondrej.Cepek@mff.cuni.cz

Abstrakt: Práca sa zaoberá časovou zložitosťou problému minimalizácie formúl reprezentujúcich Booleovské funkcie. Prezentuje základné koncepty z oblastí Booleovských funkcií, ich zápisu v normálnych formách a minimalizácie týchto zápisov. Celá kapitola je venovaná Umansovym [13] dôkazom Σ_2 úplností minimalizácie DNF formúl pre obe základné používané miery minimality všeobecných funkcií. Pre triedu formulí nazývané Matched prináša nové výsledky, ktoré ukazujú, že aj keď je pre Matched formule jednoduchý problém splniteľnosti (ťažký pre všeobecné formule), problémy spojené s minimalizáciou a aj samotná minimalizácia je pre ne rovnako ťažká ako pre všeobecné formule.
Kľúčové slová: Booleovské funkcie, Booleovská minimalizácia, Matched formuly

Title: Time complexity of Boolean minimization.
Author: Štefan Gurský
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: doc. RNDr. Ondřej Čepek, Ph.D.
Supervisor's e-mail address: Ondrej.Cepek@mff.cuni.cz

Abstract: This thesis deals with the time complexity of Boolean minimization – minimization of formulae that represent Boolean functions. It presents basic concepts from the area of Boolean functions, of their normal form representations and of minimization of these representations. A whole chapter is dedicated to Umans's [13] proofs of Σ_2 completeness of minimization of general DNF formulae for both common measures of minimality. For a class of formulae called Matched this thesis presents new results that show that although satisfiability problem is easy for Matched formulae (difficult for an arbitrary formula), problems connected to minimization and minimization itself is as hard for Matched formulae as it is for general formulae.
Keywords: Boolean functions, Boolean minimization, Matched formulae

Kapitola 1

Úvod

Booleovské funkcie predstavujú zaujímavý konštrukt. Na prvý pohľad vyzerajú ako jednoduchá a ľahko pochopiteľná myšlienka, nad ktorej skúmaním je nemožné stráviť viac ako niekoľko hodín, no napriek tomu sa ich štúdiu venujú tímy matematikov a informatikov po celom svete už niekoľko desaťročí.

Booleovské funkcie sú spojené najmä s oblasťou zložitosti, kde problémy týkajúce sa Booleovských funkcií sú často úplnými pre rôzne triedy zložitosti. Aj prvý problém, ktorého NP úplnosť bola dokázaná, je z oblasti Booleovských funkcií.

Pre praktické používanie Booleovských funkcií je vhodné mať ich reprezentované v nejakom štandardnom tvare, pričom ich zápis by mal byť čo najmenší. Táto práca sa zaoberá časovou zložitosťou minimalizácie Booleovských funkcií zapísaných v niektorej z normálnych foriem a tiež špeciálnej podtriedy Matched formúl.

Prvá kapitola prináša definíciu Booleovských funkcií a popisuje spôsob ich zápisu. V druhej kapitole predstavíme normálne formy a ukážeme niektoré ich vlastnosti. Tretia kapitola predstaví niektoré triedy zložitosti problémov, do ktorých problémy týkajúce sa minimalizácie patria. Štvrtá kapitola popisuje problém minimalizácie všeobecne a na príkladoch niektorých funkcií.

Piata kapitola prezentuje dva základné dôkazy z oblasti minimalizácie Booleovských funkcií v DNF a to Σ_2 úplnosť tejto minimalizácie pri oboch bežne používaných mierach.

Šiesta kapitola obsahuje popis triedy formúl zvanej Matched. Pre tieto formuly nebol problém minimalizácie preskúmaný. Šiesta kapitola prináša

niektoré zaujímavé zistenia o Matched formulách – aj keď je pre nich problém splniteľnosti jednoduchý, problémy spojené s minimalizáciou sú na nich rovnako ťažké ako na ostatných formulách.

1.1 Booleovské funkcie

Definícia (Booleovská funkcia)**.** Booleovská funkcia je zobrazenie z množiny $\{0,1\}^n$ do množiny $\{0,1\}$.

Alternatívne sa dá Booleovská funkcia zadefinovať ako zobrazenie z vektorového priestoru $\mathbb{Z}_2^n$ do $\mathbb{Z}_2$.

Poznámka 1.1.1. Hodnotám jedna a nula sa často udeľujú pravdivostné hodnoty *True* resp. *False*. O funkcii povieme, že je pravdivá pri nejakom ohodnotení premenných, ak jej hodnota pri tomto ohodnotení je jedna.

Definícia (Truepoint)**.** Truepoint (pravdivý bod) nazveme ohodnotenie premenných, na ktorom má funkcia hodnotu jedna.

Definícia (Falsepoint)**.** Falsepoint (nepravdivý bod) nazveme ohodnotenie premenných, na ktorom má funkcia hodnotu nula.

Booleovská funkcia je podľa definície funkciou na konečnej množine [1]. Z toho vyplýva, že je možné ju zapísať tabuľkou. Takej tabuľke sa hovorí pravdivostná tabuľka. Takáto tabuľka je jedným z možných zápisov Booleovskej funkcie. Pravdivostná tabuľka má nevýhodu, že jej veľkosť rastie exponenciálne s počtom premenných a preto sa rýchlo stáva príliš veľkou. Pre každú funkciu na n premenných obsahuje 2^n riadkov. Keďže každý z 2^n vstupov môže nadobúdať jednu z hodnôt nula alebo jedna, je zrejmé, že funkcií na n premenných je 2^{2^n}.

V niektorých prípadoch sa ale funkcia dá zapísať úspornejšie ako tabuľkou.

[1] Niektoré funkcie sa dajú prirodzene rozšíriť na ľubovoľný počet premenných (napríklad funkcia, ktorá vráti jedničku práve keď vstup obsahuje tri jedničky). Na počte vstupov u nej nezáleží.

1.2 Logické spojky

Jedným s obľúbených spôsobov zapisovania Booleovských funkcií je skladanie z niekoľkých základných funkcií zvaných *logické spojky*[2]. Najčastejšie používanými spojkami sú spojky $\{\neg, \wedge, \vee\}$ – negácia, spojka „a“ a spojka „alebo“ (označované tiež anglickými názvami „NOT“, „AND“ a „OR“).

Tabuľka 1.1: Najbežnejšie logické spojky

(a) NOT

x	$\neg x$
0	1
1	0

(b) AND

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

(c) OR

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Pri používaní spojok AND, OR a NOT je možné stretnúť sa so skráteným zápisom. Negácia sa v tom prípade značí vodorovnou čiarou nad negovanou funkciou (vo väčšine prípadov použitia skráteného zápisu ide len o negáciu jednej premennej), spojka AND sa označuje rovnako ako násobenie bez symbolu a spojka OR ako sčítanie. Priority spojok sa v tomto prípade správajú rovnako ako pri sčítaní a násobení.

Príklad. Funkcia $(\neg a \wedge b) \vee (\neg c \wedge \neg d)$ by sa zapísala skrátene ako $\overline{a}b + \overline{c}\overline{d}$.

Pomocou týchto spojok (spolu s konštantnými funkciami) je možné vytvoriť ľubovoľnú Booleovskú funkciu. Množina spojok s takouto vlastnosťou sa nazýva úplná báza. $\{0, 1, \neg, \wedge, \vee\}$ nie je jediná možná úplná báza, ale je najpoužívanejšia. Inými používanými spojkami, sú ešte implikácia ($\Longrightarrow$), ekvivalencia ($\Longleftrightarrow$), XOR ($\oplus$), NAND a NOR. Aby spojky boli čo najjednoduchšie, sú to funkcie na dvoch premenných (okrem spojky NOT). Zo šestnástich možných funkcií na dvoch premenných ich má polovica svoj názov.

[2]K nim sa ešte pridávajú konštantné funkcie

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

(a) XOR

x	y	x NAND y
0	0	1
0	1	1
1	0	1
1	1	0

(b) NAND

x	y	x NOR y
0	0	1
0	1	0
1	0	0
1	1	0

(c) NOR

Tabuľka 1.2: Iné pomenované spojky

1.3 Vlastnosti niektorých spojok

Zo spomenutých spojok nás budú zaujímať hlavne spojky AND, OR a NOT. Tie stačia na zapísanie každej booleovskej funkcie. [3]

Dvakrát použitá spojka NOT je identita, preto dve za sebou idúce spojky NOT môžeme zo zápisu funkcie vždy odstrániť.

Spojky AND a OR sú obidve komutatívne aj asociatívne. To znamená, že ich môžeme jednoducho rozšíriť na neobmedzené množstvo premenných, pričom AND bude mať hodnotu jedna práve keď všetky vstupy budú mať hodnotu jedna a OR bude mať hodnotu jedna práve keď aspoň jeden vstup bude mať hodnotu jedna.

Medzi spojkami AND, OR a NOT platia nasledujúce vzťahy:

Distributivita:

$$a \vee (x \wedge y) = (a \wedge x) \vee (a \wedge y)$$
$$a \wedge (x \vee y) = (a \vee x) \wedge (a \vee y)$$

De Morganove pravidlá:

$$\neg(x \wedge y) = (\neg x \vee \neg y) \tag{1.1}$$
$$\neg(x \vee y) = (\neg x \wedge \neg y) \tag{1.2}$$

Tieto pravidlá sú užitočné pri prevode zápisu funkcie na iný ekvivalentný zápis a dajú sa prirodzene rozšíriť na väčší počet premenných.

Spojka XOR je tiež asociatívna a komutatívna. Tým sa tiež dá rozšíriť na iný počet premenných ako dve. Jej výstupom je jednička práve keď počet jedničiek na vstupoch je nepárny. Takáto funkcia sa nazýva parita a vďaka niektorým svojim vlastnostiam je dôležitou funkciou.

[3] Pozorovanie 2.1.1 v kapitole 2.

Spojky NAND a NOR sú každá samostatná – každá sama o sebe je úplnou bázou. Tieto spojky sú komutatívne, ale už nie sú asociatívne. Preto sa na viac premenných rozširujú ako negácie spojok AND a OR. Z toho aj vznikli ich názvy. Tieto spojky majú tiež svoje staré názvy (NAND je Shefferov operátor, NOR je Peirceova šípka [6]) a aj značky (NAND má šípku nahor $\uparrow$, NOR má šípku nadol $\downarrow$). Hradlá reprezentujúce NAND a NOR sa používajú v integrovaných obvodoch, pretože sú lacné a dá sa nimi reprezentovať každá funkcia. Spojkami NAND a NOR sa ďalej v práci nebudeme zaoberať.

Kapitola 2

Normálne formy

2.1 Základné definície

Pomocou spojok AND, OR a NOT sa dá zapísať ľubovoľná Booleovská funkcia mnohými spôsobmi. Dva spôsoby zápisu ale používané výrazne častejšie ako ostatné. Tým sa hovorí normálne formy. Poznáme disjunktívnu normálnu formu a konjunktívnu normálnu formu. Kým sa však dostaneme k ich popisu, je vhodné zadefinovať si niektoré pojmy.

Definícia (Literál)**.** Literál je premenná alebo jej negácia. Literál x s literálom $\overline{x}$ sa nazývajú komplementárny pár.

Definícia (Term)**.** Term je konjunkcia literálov – literály spojené spojkou AND – v ktorých sa nevyskytuje komplementárny pár.

Keďže spojka AND je rozšíriteľná na ľubovoľné množstvo premenných, dá sa na term pozerať aj ako na množinu literálov (ktoré sú spojené spojkou AND) a opačne. Podmnožina termu je potom term na podmnožine literálov obsiahnutých v terme. Prázdny term má hodnotu jedna.

Definícia (Klauzula)**.** Klauzula je disjunkcia literálov – literály spojené spojkou OR.

Pre klauzulu platí to, čo pre term – dá sa na ňu pozerať ako na množinu literálov. Prázdna klauzula má hodnotu nula.

Definícia (Dĺžka klauzule/termu)**.** Dĺžkou klauzule/termu nazveme počet literálov v klauzule/terme.

Definícia (Minterm)**.** Pre Booleovskú funkciu na n premenných je minterm taká konjunkcia literálov v ktorej sa každá premenná nachádza práve raz.

Definícia (Maxterm)**.** Pre Booleovskú funkciu na n premenných je maxterm disjunkcia literálov v ktorej sa každá premenná nachádza práve raz.

Mintermov aj maxtermov je zrejme 2^n. Minterm je pravdivý práve na jednom ohodnotení vstupov. Maxterm je naopak práve na jednom ohodnoteni nepravdivý.

Definícia (Konjunktívna normálna forma)**.** Konjunktívna normálna forma je konjunkcia klauzúl.

Definícia (Disjunktívna normálna forma)**.** Disjunktívna normálna forma je disjunkcia termov.

Konjunktívna normálna forma sa skracuje na CNF a disjunktívna na DNF. V ďalšom texte budú tieto skratky často používané.

Pozorovanie 2.1.1. Každá Booleovská funkcia sa dá zapísať aj v DNF aj v CNF.

Dôkaz. Zapíšme si funkciu pravdivostnou tabuľkou. Pre DNF si vyberme riadky, v ktorých je hodnota funkcie jedna. Pre každý tento riadok vyrobme minterm a to tak, aby hodnota tohto mintermu by na danom vstupe bola jedna – pri ohodnoteni premennej x nulou použijeme literál $\neg x$, pri ohodnotení jedničkou literál x. Disjunkcia týchto mintermov bude DNF pre zadanú funkciu. Pre každý vstup, kde bola funkcia jedničková je jedničkový práve jeden term, preto je aj ich disjunkcia jedničková. Pre vstup, kde bola funkcia nulová nie je jedničkový žiadny minterm a preto aj disjunkcia bude nulová.

Pre CNF je dôkaz analogický. Vytvoríme si maxtermy nepravdivé na vstupoch, na ktorých je funkcia nulová a ich konjunkcia bude CNF pre danú funkciu.

Konštantnú funkciu zapíšeme priamo jej hodnotou. □

Poznámka 2.1.2. Týmto je tiež dokázané, že každá funkcia sa dá zapísať za pomocí spojok AND, OR a NOT. V prípade, že nechceme povoliť konštanty, je možné nulu nahradiť funkciou $x \wedge \neg x$ a jedničku funkciou $x \vee \neg x$.

Poznámka 2.1.3. Formálne by sa malo rozlišovať medzi funkciou ako zobrazením a jej zápisom (formulou), či už v CNF alebo DNF, alebo v inom tvare. Kde však nebude hroziť nejasnosť, bude miesto zápisu „funkcia reprezentovaná formulou φ“ používaný zápis „funkcia φ“. Ekvivalencia dvoch a viacerých formulí bude znamenať, že reprezentujú tú istú funkciu.

2.2 Podobnosti a vzťahy medzi DNF a CNF

Zápisy a terminológia okolo DNF a CNF formulí sa na seba v mnohom podobajú. V oboch prípadoch sa dá na zápis formuly pozerať ako na zápis množiny termov resp. klauzúl, pričom tieto sa dajú reprezentovať ako množiny literálov.

Pozorovanie 2.2.1. Ak máme známu DNF φ reprezentujúcu funkciu f, CNF reprezentujúca funkciu $\neg f$ sa vytvorí pridaním negácie pred každú premennú a nahradením spojok AND za OR a spojok OR za AND.

Definícia (Duálna funkcia)**.** Duálna funkcia k funkcii $f(x)$ je funkcia $f^* = \neg f(\neg x)$, kde $\neg x$ je doplnok vstupu.

Pozorovanie 2.2.2. Podobne ako v predchádzajúcom pozorovaní vďaka De Morganovym vzťahom pre DNF φ reprezentujúcu f vytvoríme CNF ψ funkcie f^* prepísaním spojok AND na OR a spojok OR na AND.

Tieto dve pozorovania dovoľujú mnoho dôkazov tvrdení pre jednu z normálnych foriem použiť pre rovnaký alebo podobný problém pri druhej normálnej forme.

Definícia (Implikant)**.** Implikant C pre Booleovskú funkciu φ je taký term C, pre ktorý platí $C \implies \varphi$, teda ak je C pravdivý, tak aj φ musí byť pravdivá.

Definícia (Implikát)**.** Implikát C pre Booleovskú funkciu φ je taká klauzula, pre ktorú platí $\varphi \implies C$, teda ak je φ pravdivá, tak aj C musí byť pravdivá.

Poznámka 2.2.3. Každý term v DFN je implikantom funkcie reprezentovanej danou DNF a každá klauzula v CNF je implikátom funkcie reprezentovanej danou CNF. Implikát a implikant majú podobne znejúce názvy a často sú medzi sebou zameňované. Základnou pomôckou na odlíšenie je párovanie implikát $\leftrightarrow$ CNF a implikant $\leftrightarrow$ DNF.

Ku termom a implikantom sa viažu čiastočné ohodnotenia. Čiastočné ohodnotenie premenných je priradenie hodnôt podmnožine premenných danej funkcie.

Pozorovanie 2.2.4. Medzi termami a čiastočnými ohodnoteniami existuje bijekcia.

Dôkaz. Pre term zostrojíme čiastočné ohodnotenie tak, že premennej, ktorá je v terme pozitívne priradíme jedničku a premennej, ktorá je v terme negatívne priradíme nulu. Ostatné premenné necháme neohodnotené. Čiastočnému ohodnoteniu vyrobíme term tak, že premennú ohodnotenú jedničkou dáme do termu pozitívne a premennú ohodnotenú nulou negatívne. Neohodnotené premenné do termu nezahrnieme. □

Definícia (Pokrývanie)**.** O terme t povieme, že pokrýva ohodnotenie A práve vtedy keď A je rozšírením čiastočného ohodnotenia príslušného termu.

Je vidieť, že term pokrýva práve tie ohodnotenia, v ktorých je pravdivý. Pokrývanie takto môžeme rozšíriť aj na funkcie. Funkcia pokrýva tie ohodnotenia, na ktorých je pravdivá. Nasledujúce pozorovanie je zrejmé.

Pozorovanie 2.2.5. Term t je implikantom f ak všetky ohodnotenia, ktoré pokrýva sú truepointami f.

Definícia (Konzistencia čiastočných ohodnotení)**.** Dve čiastočné ohodnotenia sú konzistentné ak neexistuje premenná, ktorej by jedno priradilo nulu a druhé jedničku.

Definícia (Absorpcia)**.** Hovoríme, že implikant C' funkcie φ absorbuje implikant C ak množina literálov reprezentovaná implikantom C' je podmnožinou množiny literálov, ktorá je reprezentovaná implikantom C.

Analogicky pre implikát: implikát C' absorbuje implikát C práve keď množina literálov v C' je podmnožinou literálov v C.

Poznámka 2.2.6. Keďže na termy aj klauzule sa dá pozerať ako na množiny literálov, má zmysel nazývať jeden term podmnožinou iného termu. Implikant (implikát) potom absorbuje tie implikanty (implikáty), ktorých je podmnožinou. Nazývať však klauzulu podmnožinou nejakého termu a naopak pravdepodobne zmysel nemá.

Definícia (Primárny implikant/implikát)**.** Primárny implikant pre funkciu φ je taký implikant funkcie φ, že žiadna vlastná podmnožina jeho literálov už nie je implikantom funkcie φ.

Analogicky primárny implikát je taký implikát funkcie φ, ktorého žiadna vlastná podmnožina nie je implikát.

Poznámka 2.2.7. Pre tautológiu je primárnym implikantom prázdny term. Pre kontradikciu je zasa prázdna klauzula implikátom. Týmto je tiež vysvetlená voľba považovať prázdny term za jedničku a prázdnu klauzulu za nulu.

V niekoľkých nasledujúcich odsekoch sa zameriame na implikanty a DNF. Analogické tvrdenia ale platia aj o implikátoch a CNF.

Lema 2.2.8. Majme funkciu f reprezentovanú DNF φ a jej implikant C. Potom $\varphi \vee C$ reprezentuje f.

Dôkaz. Ak je φ rovné jedničke, potom je zrejme aj $\varphi \vee C$ rovné jedničke. Ak je φ rovné nule, potom by $\varphi \vee C$ mohlo byť nerovné nule iba v prípade, že C má hodnotu jedna. Stav, keď φ má hodnotu nula, ale C má hodnotu jedna je ale v spore s tým, že C je implikantom φ. φ je teda ekvivalentné s $\varphi \vee C$ a obe reprezentujú funkciu f. □

Pozorovanie 2.2.9. Majme termy C a C', ktorý ho absorbuje. C je implikantom C'.

Dôkaz. Splnením termu C splníme všetky literály termu C'. □

Lema 2.2.10. Majme DFN $\varphi = t_1 \vee t_2 \vee \ldots \vee t_n$. Ak C je implikant φ absorbujúci t_n, DNF $\varphi' = t_1 \vee t_2 \vee \ldots t_{n-1} \vee C$ je ekvivalentná φ.

Dôkaz. Označme si $\psi = t_1 \vee t_2 \vee \ldots \vee t_{n-1}$. Potom $\varphi \equiv \psi \vee t_n$ a $\varphi' \equiv \psi \vee C$. Ak je φ rovná jednej, potom buď ψ je rovná jednej a teda aj φ' je rovná jednej, alebo je t_n rovný jednej a podľa pozorovania 2.2.9 je aj C rovný jednej, teda φ' je rovná jednej. Opačným smerom: Ak je φ' rovná jednej, tak buď je ψ rovná jednej a potom je aj φ rovná jednej, alebo je C rovný jednej, ale keďže je implikantom φ, musí byť φ rovná jednej. □

Veta 2.2.11. Funkcia sa dá zapísať ako disjunkcia všetkých svojich primárnych implikantov. Táto DNF sa nazýva kanonická DNF danej funkcie.

Dôkaz. Majme funkciu f, ktorú chceme zapísať ako DNF zloženú zo všetkých primárnych implikantov. Vezmeme si nejakú DNF φ, ktorá reprezentuje f. Pre každý jej term nájdeme primárny implikant, ktorý ho absorbuje (z definície – prejdeme podmnožiny termu, vyberieme tie, ktoré sú implikantom f a z nich vyberieme jeden v inklúzii minimálny) a nahradíme term týmto primárnym implikantom. Podľa 2.2.10 je funkcia reprezentovaná touto zmenenou DNF stále f. Nájdeme všetky ďalšie primárne implikanty f (termov na premenných z f je konečne mnoho) a pridáme ich k DNF, ktorú už máme. Podľa lemy 2.2.8 je výsledná DNF stále reprezentáciou funkcie f. □

Poznámka 2.2.12. Až na poradie jednotlivých termov a literálov v nich je kanonická DNF pre funkciu určená jednoznačne.

Poznámka 2.2.13. Spôsob hľadania kanonickej DNF postupom z dôkazu vety 2.2.11 je síce funkčný a vždy skončí, ale je až príliš neefektívny. Nepoužíva sa.

2.3 Hľadanie všetkých primárnych implikantov

Kanonická DNF môže slúžiť ako istý normovaný zápis pre Booleovskú funkciu. Je preto dobre vedieť ju nájsť. Jeden zo spôsobov je prejsť všetky termy na premenných funkcie, zistiť, ktoré z nich sú primárnymi implikantami a tie použiť v kanonickej DNF.

Iným spôsobom hľadania kanonickej DNF je konsenzuálna metóda, ktorá na vstupe dostane DNF a upravuje ju, kým nezíska kanonickú.

Definícia (Konflikt)**.** O termoch t_1 a t_2 hovoríme, že majú konflikt v premennej x ak t_1 obsahuje literál x a t_2 literál $\overline{x}$ alebo naopak.

Definícia (Konsenzus)**.** O termoch t_1 a t_2 hovoríme, že majú konsenzus ak majú konflikt v práve jednej premennej. Ak $t_1 = Ax$ a $t_2 = B\overline{x}$, ich konsenzus je term AB.

Lema 2.3.1. Konsenzus dvoch implikantov funkcie f je tiež implikantom funkcie f.

Dôkaz. Nech prvý implikant je tvaru Ax a druhý tvaru $B\overline{x}$, pričom A a B nemajú konflikt. Tvrdíme, že AB je tiež implikantom f. Predpokladajme, že nie je. V tom prípade existuje ohodnotenie, ktoré AB nastaví na jedničku, ale funkciu f na nulu. A aj B musia byť týmto ohodnotením obe nastavené na jedničku. Toto ohodnotenie tiež nejakým spôsobom nastaví premennú x. Ak ju nastaví na nulu, pravdivým sa stane implikant $B\overline{x}$ a f bude musieť byť pravdivá, čo je spor. Ak ju nastaví na jedničku, pravdivým sa stane implikant Ax a opäť bude f musieť byť pravdivá. AB je teda implikantom f. □

Predchádzajúcu lemu môžeme využiť na generovanie implikantov pomocou konsenzuálnej metódy.

Definícia (Konsenzuálna metóda)**.** Konsenzuálnou metódou nazývame nasledujúci algoritmus na nájdenie primárnych implikantov. Vstupom je nejaká DNF φ a výstupom je kanonická DNF ekvivalentná so vstupnou.

1. Kým vo φ existuje nejaký term t absorbovaný iným termom t' z φ, vyhoď t z φ.

2. Nájdi dva termy, ktorých konsenzus nie je absorbovaný žiadnym termom z φ a pridaj ho do φ. Ak takéto termy neexistovali, skonči. Ináč pokračuj prvým bodom.

Veta 2.3.2. Konsenzuálna metóda vráti kanonickú DNF pre funkciu reprezentovanú jej vstupom.

Dôkaz. Podľa lemy 2.3.1, lemy 2.2.8 a lemy 2.2.10 sa počas celého behu algoritmu funkcia reprezentovaná medzivýsledkami nelíši od funkcie na vstupe. Preto výstup bude reprezentovať tú istú funkciu ako vstup.

Použijeme dôkaz z [9], aby sme ukázali, že vygenerované boli všetky primárne implikanty. Označíme si výstup z konsenzuálnej metódy $\varphi = T_1 \vee T_2 \vee \ldots \vee T_n$ a pre spor predpokladajme, že existuje primárny implikant T, ktorý vo φ chýba[1]. T zrejme obsahuje iba premenné z φ, pretože ostatné (na ktorých nezávisí) by sme mohli vynechať a stále by sme mali implikant. Keďže T je primárny implikant ale nie je vo φ, neexistuje i, pre ktoré by T_i absorboval T. Existuje aspoň jeden term (napríklad samotný term T), ktorý spĺňa nasledovné tri podmienky: (a) je absorbovaný termom T, (b) nie je absorbovaný žiadnym termom z φ a (c) obsahuje iba premenné z φ. Nech S je spomedzi týchto termov ten najdlhší. Ak by S obsahoval všetky premenné z φ, mal by konflikt s každým termom T_i [2], takže φ by bola v (čiastočnom) ohodnotení prilúchajúcom termu S nulová. S ale implikuje T, čo je implikát φ, takže tá nemôže byť nulová pri pravdivom S – spor. Term S teda neobsahuje všetky premenné. Nech neobsahuje premennú x. Keďže S je najdlhší term splňujúci (a), (b) a (c), termy Sx a $S\overline{x}$ nespĺňajú bod (b) (body (a) a (c) zrejme spĺňajú). Termy Sx a $S\overline{x}$ sú každý absorbované nejakým termom z φ, pričom jeden z týchto termov obsahuje x a druhý $\overline{x}$, keďže S nebol absorbovaný samostatne. Nech jeden z týchto termov je Ax a druhý $B\overline{x}$. A aj B absorbujú S, takže Ax s $B\overline{x}$ majú konsenzus AB. Tento

[1] V pôvodnom dôkaze sa predpokladalo, že je neprázdny, pretože sa o prázdnych termoch neuvažovalo.

[2] Okrem prázdneho termu – v tom prípade by ale celá DNF bola tvorená len týmto jedným termom.

konsenzus absorbuje S a nie je absorbovaný žiadnym termom T_i, pretože S nebolo absorbované žiadnym termom T_i. Algoritmus konsenzuálnej metódy teda nemohol zastaviť s φ ako výstupom, pretože ešte mohol pridať term AB.

Ukázali sme, že ak konsenzuálna metóda zastaví, v jej výstupe budú všetky primárne implikanty. Dokázať, že zastaví je jednoduché. Ak sa do φ pridá nejaký term, už nikdy sa do nej nepridá zas (buď tam ostane a bude blokovať ďalšie pokusy o opakované pridanie alebo bude absorbovaný a term, ktorý ho absorboval bude tiež blokovať pokusy o opakované pridanie). Algoritmus zastavuje ak sa v druhom kroku nepridá žiaden term. Keďže termov je len konečne mnoho a žiaden sa nepridá dvakrát, algoritmus zastaví.

Poslednou vecou, ktorá zostáva, je ukázať, že vo výstupe nebude nič okrem primárnych implikantov. Už vieme, že vo výstupe budú všetky primárne implikanty. Všetky ostatné implikanty ale obsahujú nejaký primárny implikant, ktorý ich absorbuje a preto použitím prvého pravidla konsenzuálnej metódy museli byť vyhodené. □

Kapitola 3

Triedy zložitosti

3.1 Ľahké problémy

V ďalšom texte budú spomínané rôzne triedy problémov a prevody medzi problémami. Problémy si predstavíme ako úlohy riešené programami na počítači. Program rieši nejaký problém na dátach. Dáta nazveme inštanciou problému. Klasickým príkladom z oblasti Booleovských funkcií je splniteľnosť. Problém je: „Je zadaná formula splniteľná?“ Môžeme mať program, ktorý to zisťuje a ako vstup mu dávame rôzne formuly a program nám odpovedá. Formula bude inštanciou. Problém splniteľnosti je problémom, ktorý sa objavuje v rôznych obmenách v celej teórii zložitosti a ešte sa k nemu niekoľkokrát vrátime.

Je samozrejme možné mať program, ktorý rozhoduje splniteľnosť iba jednej formuly, ktorú má v sebe zakódovanú. Tento program však po jednom behu môžeme nahradiť programom, ktorého jediným účelom bude vypisovať zistenú odpoveď. Spúšťať pôvodný program znova nebude mať význam, pretože nám vypíše odpoveď, ktorú už aj tak poznáme, a bude mu to trvať dlho. No to, že mu to bude trvať dlho bude vlastnosťou programu a nie vlastnosťou problému. Po jednom behu programu sa problém stane triviálnym. My sa budeme zaoberať zložitosťami problémov nezávisle na tom, aký program na riešenie použijeme. Budeme preto potrebovať, aby vstupov do programu mohlo byť nekonečne mnoho.

Problémy, ktorými sa budeme zaoberať, budú rozhodovacie. Odpoveďou bude vždy buď áno, alebo nie.

Definícia (Pozitívna/negatívna inštancia)**.** Pokiaľ na inštanciu problému je odpoveď áno, nazveme túto inštanciu pozitívnou. V opačnom prípade

negatívnou.

Budeme potrebovať nejakým spôsobom zisťovať, koľko prostriedkov sme použili na riešenie nejakého problému v závislosti na obtiažnosti vstupu. Počítať, ako dlho bežal program na počítači síce možné je, ale počítače sa zrýchľujú. Preto sa používajú rôzne výpočtové modely.

Známym výpočtovým modelom je Turingov stroj. Definíciu je možné nájsť v každej knihe zaoberajúcej sa výpočtovou zložitosťou, napríklad v [1]. Ide o zariadenie, ktoré má pamäť (reprezentovanú nekonečnou páskou), vstup (zapísaný na páske v nejakej konečnej abecede), inštrukcie (opäť konečné) a výstup. Podľa charakteru výstupu rozoznávame rozhodovacie stroje (akceptory), ktoré vracajú jednobitový výstup oznamujúci, či je vstup prijatý alebo neprijatý, a prepisovacie stroje, ktoré vracajú ako výstup slovo na páske (transducery). Iné delenie je na deterministické a nedeterministické stroje, kde deterministické majú v každom kroku najviac jednu inštrukciu, ktorú môžu použiť (a ak nemajú žiadnu, tak sa zastavia) a nedeterministické, u ktorých je možné mať viacero inštrukcií použiteľných v jednej situácii (stroj si potom „magicky“ vyberá tú, ktorá vedie k riešeniu). Ak nie je povedané ináč, Turingovým strojom budeme myslieť jeho deterministický variant.

Pre rozhodovacie problémy budeme používať akceptory (transducery využijeme pri prevodoch problémov). Dôležité je, že pre rôzne problémy máme rôzne stroje (Turingov stroj teda funguje ako program) a inštancia sa mu dodá zapísaná v nejakej abecede. Turingov stroj sa tu môže považovať za zariadenie, ktoré potom zisťuje, či inštancia je pozitívna alebo nie. Rozhodovacie problémy sa často stotožňujú s jazykmi, pretože Turingov stroj rozhoduje náležanie do jazyka.

Turingov stroj na to, aby mohol rozhodnúť o tom, či je inštancia pozitívna alebo nie potrebuje určitý čas. Tento čas je závislý najmä na veľkosti vstupu (inštancie). Podľa tohto času sa potom dajú problémy deliť na zložité a menej zložité.

Za „nie príliš zložité“ sa považujú problémy, ktorých čas riešenia rastie pomalšie ako nejaký polynóm závislý na dĺžke vstupu. Problémom, ktoré sa takto dajú riešiť sa hovorí „problémy v P“.

Definícia (P)**.** Trieda P je trieda problémov (jazykov), pre ktoré existuje (deterministický) Turingov stroj, ktorý ich rozhoduje v čase, ktorý sa dá zhora ohraničiť polynómom od dĺžky vstupu.

3.2 Ťažšie problémy

Ďalšiou dôležitou triedou problémov je trieda NP. To sú problémy, ktoré sa samy o sebe pravdepodobne nedajú vyriešiť v polynomiálnom čase, ale ak niekto zistí, že inštancia je kladná, „nebude mať problém nás o tom presvedčiť“ – budeme schopní v polynomiálnom čase jeho riešenie overiť ak nám dodá „rozumné množstvo“ ďalšej informácie. O tom, či budeme schopní v polynomiálnom čase overiť aj negatívnu inštanciu sa nič netvrdí.

Definícia (NP)**.** NP je trieda problémov/jazykov, kde pre pozitívnu inštanciu existuje najviac polynomiálne veľká dodatočná informácia (certifikát), ktorá pridaná k danej inštancii dovolí v polynomiálnom čase overiť pozitivitu inštancie.

Poznámka 3.2.1. NP sa často definuje pomocou nedeterministického Turingovho stroja (odtiaľ N v názve) bežiaceho v polynomiálnom čase. Ten prijíma slovo ak existuje prijímajúci výpočet, čo je v súlade s požiadavkom existencie certifikátu.

Poznámka 3.2.2. Už spomenutý problém splniteľnosti zrejme do NP patrí. Ak získame ohodnotenie, na ktorom je formula pravdivá, je už jednoduché (v P) splnenie overiť.

Definícia (coNP)**.** coNP je trieda problémov/jazykov, kde pre negatívnu inštanciu existuje certifikát ako v definícii 3.2.

Poznámka 3.2.3. Problém v coNP je negáciou problému v NP.

Poznámka 3.2.4. coNP sa dá definovať tak, že pri pozitívnej inštancii pre všetky možné certifikáty vieme overiť, že certifikát nepotvrdí negatívnu inštanciu.

Príklad. Ako príklad si uvedieme negáciu problému splniteľnosti. Inštanciou bude formula a problémom (otázkou): Je formula nesplniteľná? Pre odpoveď „nie“ existuje certifikát (splňujúce ohodonotenie). Pre odpoveď „áno“ musíme skontrolovať všetky možné ohodnotenia (certifikáty) a ak ani jeden z nich nepotvrdí odpoveď „nie“, teda všetky ohodnotenia sú nesplňujúce, môžeme formulu prehlásiť za nesplniteľnú.

Predpokladá sa, že problémy, ktoré sú jednoducho riešiteľné (P), sú vlastnou podmnožinou problémov, ktorých pozitívne resp. negatívne inštancie sú

jednoducho overiteľné (NP, resp. coNP), ale problém je doteraz nevyriešený a aj keď sa názory rôznia, väčšina informatikov si myslí, že P je rôzne od NP [3].

Tried problémov je mnoho. Veľkú väčšinu je možné nájsť na stránkach Complexity Zoo (`http://qwiki.stanford.edu/wiki/Complexity_Zoo`).

3.3 Prevody a úplnosť

V niektorých prípadoch je možné získať riešenie problému tak, že ho upravíme a vyriešime nový problém. Takýto postup sa nazýva prevod problému. Je vhodné, aby prevod nebol zložitejší, ako vyriešenie pôvodného problému. Problém, na ktorý sa prevádza je potom aspoň tak ťažký, ako pôvodný problém.

Prevodov existuje niekoľko druhov, ale pre nás najzaujímavejší je prevod v polynomiálnom čase, pretože polynomiálny čas považujeme za dostatočne jednoduchý.

Definícia (Prevod problému)**.** Problém A je previediteľný na problém B pokiaľ existuje deterministicky Turingov stroj typu transducer, pracujúci v polynomiálnom čase, ktorý pre pozitívnu inštanciu problému A vyrobí pozitívnu inštanciu problému B a pre negatívnu inštanciu problému A vyrobí nagatívnu inštanciu problému B.

Pre triedu problémov tak môžeme hľadať problémy, ktoré sú v danej triede najťažšie. Takéto problémy nazývame úplnými pre danú triedu.

Definícia (NP úplný problém)**.** Problém nazveme NP úplný ak patrí do triedy NP a zároveň každý problém z NP sa dá na tento problém v polynomiálnom čase previesť (o probléme v tom prípade hovoríme, že je NP ťažký).

Príklad. Problém splniteľnosti je NP úplný. Je to prvý problém, o ktorom sa vedelo, že je NP úplným.

NP úplná je dokonca nasledujúca verzia problému:

SAT

Inštancia: CNF φ

Otázka: Existuje ohodnotenie, ktoré splní funkciu reprezentovanú φ?

Poznámka 3.3.1. Pre DNF problém splniteľnosti ťažký nie je – stačí splniť ľubovoľný z termov a na jeho splnenie použijeme čiastočné ohodnotenie, ktoré term reprezentuje. Pre DNF je ťažký duálny problém – zistiť, či zadaná DNF má niekedy hodnotu nula, teda či je falzifikovateľná. Tento problém sa nazýva „FALS“. Jeho doplnok (zistiť, či daná DNF je tautológia) je potom zrejme coNP úplný (pozri nižšie) a nazýva sa „TAUT“.

Analogicky môžeme definovať problém úplný pre triedu coNP a iné triedy. Pre triedu coNP sú úplné negácie NP úplných problémov. Preto na ukázanie coNP úplnosti problému A stačí previesť NP úplný problém na negáciu problému A alebo previesť negáciu NP úplného problému na problém A. Pre triedu P nemá takáto definícia príliš zmysel, pretože v rámci prevodu by sme mohli pôvodný problém vyriešiť. V tomto prípade sa používa prevod v logaritmickom priestore.[1]

3.4 Orákulá a hierarchia

Čo sa stane, keď k Turingovmu stroju pridáme čiernu skrinku, ktorá bude magickým spôsobom riešiť nejaký problém? Môže sa stať, že budeme schopní vyriešiť problémy rýchlejšie (v niektorých prípadoch dokonca budeme môcť vyriešiť viac problémov).

Takejto magickej skrinke hovoríme orákulum. Orákulá môžu riešiť rôzne obtiažné úlohy. Orákulum, ktoré rieši úlohu z P nám žiadne nové problémy vyriešiť nepomôže. Orákulum, ktoré rieši NP úplnú úlohu nám už ale dovolí riešiť všetky problémy z NP v polynomiálnom čase, pretože ostatné úlohy z NP prevedieme v polynomiálnom čase na NP úplnú úlohu, ktorú nám vyrieši orákulum.

Od orákula nečakáme, že sa nám bude snažiť dokázať pravdivosť svojho tvrdenia, ani že bude možné tvrdenie overiť. Preto platí nasledujúce pozorovanie:

Pozorovanie 3.4.1. Orákulum je rovnako silné ako jeho negácia. NP úplné orákulum je preto rovnako silné, ako coNP úplné orákulum.

Pomocou NP (alebo coNP) orákulí môžeme začať budovať hierarchiu známu ako polynomiálna. Vyššiu úroveň vyrobíme tak, že upravíme definíciu

[1] Úplnosť sa určuje pre triedu a typ prevoditeľnosti. V tejto práci však iný ako polynomiálny prevod nebudeme potrebovať.

NP prípadne coNP tak, že pri overovaní budeme môcť použiť orákulum pre problém úplný pre jeden stupeň hierarchie nižšie.

Definícia (Triedy Σ_k a Π_k)**.** Triedy Σ_k a Π_k sú definované induktívne:

1. Σ_0 je ekvivalentná P.
2. Π_0 je ekvivalentná P.
3. Σ_k je trieda problémov/jazykov, kde pre pozitívnu inštanciu existuje polynomiálne veľký certifiát, ktorý umožňuje overiť pozitivitu inštancie v polynomiálnom čase za použitia Π_{k-1} úplného orákula.
4. Π_k je trieda problémov, kde pre pozitívnu inštanciu ľubovoľný certifikát nepotvrdí negatívnosť inštancie, pričom potvrdenie sa dá overiť v polynomiálnom čase s Σ_{k-1} úplným orákulom.

Poznámka 3.4.2. Σ_1 je NP a Π_1 je coNP. Π_k problémy sú negáciou Σ_k problémov. Pri rozpísaní definícií pre vyššie Π alebo Σ vidíme striedanie kvantifikátorov (pre každý certifikát, existuje certifikát), pričom Π začína všeobecným, Σ začína existenčným, k určuje ich počet a na konci dostaneme niečo, čo je overiteľné v P.

Opäť sa pozrieme na problém splniteľnosti. V podobe, v akej sme ho mali uvedený sa dá chápať ako otázka: Existuje ohodnotenie (všetkých) premenných tak, aby formula bola splnená? Existenčný kvantifikátor napovedá, že problém bude patriť do Σ_1 (NP). A tento problém je skutočne Σ_1 úplný. Ak otázku upravíme tak, že pred formulu dáme k kvantifikátorov začínajúcich existenčným a spoločne pokrývajúcicich všetky premenné vo formuli, získame problém úplný pre triedu Σ_k. Takému problému sa hovorí QSAT_k. V ďalšom texte budeme potrebovať problém QSAT_2, ktorý je úplný pre Σ_2, tak si tu formálne predstavíme triedu Σ_2 a problém QSAT_2.

Jazyk/problém L patrí do Σ_2 práve keď existuje polynóm p a deterministický Turingov stroj M s NP orákulom pracujúci v čase menšom alebo rovnom p od dĺžky vstupu taký, že pre každé slovo x patriace do L (pozitívnu inštanciu) existuje certifikát P, ktorého dĺžka je menšia ako $p(|x|)$ a M prijíma jazyk (x, P) (potvrdí certifikát).

Poznámka 3.4.3. Podľa definície 3.4 má byť orákulum Π_1 úplné, ale podľa poznámky 3.4.2 je Π_1 rovné coNP a podľa pozorovania 3.4.1 je coNP úplné orákulum rovnako silné ako NP úplné orákulum.

Pre triedu Σ_2 je známy Σ_2 úplný jazyk/problém zvaný QSAT_2. V obecnom tvare vyzerá nasledovne.

QSAT_2 – obecný

Inštancia: Kvanitfikovaná Booleovská formula tvaru $\exists X_1 \forall X_2 \varphi(X_1, X_2)$ kde X_1 a X_2 sú disjunktným rozkladom množiny premenných φ.

Otázka: Je hodnota φ rovná jednej?

O probléme QSAT_2 je známe, že je Σ_2 úplný.[8]

Kapitola 4

Minimalizácia

4.1 Dôvody a miery

Už sme videli, že Booleovské funkcie sa dajú zapísať tabuľkou. Tabuľka je ale na zápis príliš veľká a nepraktická. Pre n premenných má 2^n riadkov a to aj v prípade, že funkcia bude jednoduchá (napríklad konštantná). Videli sme tiež, že funkcia sa dá zapísať v CNF alebo v DNF. V týchto prípadoch sa ale dá zapísať často viac ako jedným spôsobom. Ako príklad môže poslúžiť funkcia na troch premenných, ktorá je nepravdivá iba na vstupoch $\vec{0}$ a $\vec{1}$. V DNF sa dá zapísať ako $x\overline{y}\overline{z} + \overline{x}y\overline{z} + \overline{x}\overline{y}z + \overline{x}yz + x\overline{y}z + xy\overline{z}$ ale tiež aj ako $x\overline{y} + x\overline{z} + y\overline{x} + y\overline{z} + z\overline{x} + z\overline{y}$, ako $x\overline{y} + y\overline{z} + z\overline{x}$, ako $z\overline{y} + y\overline{x} + x\overline{z}$ a niekoľkými ďalšími spôsobmi. Prvý spôsob je pomocou všetkých mintermov, druhý pomocou všetkých primárnych implikantov a posledné dva sú zápisy pomocou najmenšieho možného počtu termov. V CNF má tá istá funkcia jediný zápis a to $(x+y+z) \wedge (\overline{x}+\overline{y}+\overline{z})$. Pri minimalizácii sa snažíme nájsť „čo najmenší“ zápis danej funkcie.

Dôvodov pre minimalizáciu je niekoľko. Matematikovi stačí estetický dôvod – kratší zápis je krajší a elegantnejší. Pre programátora sa kratší zápis jednoduchšie kontroluje. Pri uchovávaní v pamäti je zasa pre šetrenie pamäťou lepší kratší zápis. Pri malom počte funkcii na malom počte premenných je často rozdiel zanedbateľný, ale pri veľkom počte môžu byť rozdiely výrazné. Pre inžinierov tvoriacich logické obvody v hardvéri je lacnejšie implementovať funkciu za pomocou čo najmenšieho počtu hradiel.

Pri minimalizácii Booleovskej funkcie je vhodné určiť si, čo minimalizujeme. Minimalizovanú hodnotu nazveme miera. Formálne:

Definícia (Miera minimalizácie)**.** Miera je zobrazenie z Booleovských for-

mulí (nie z funkcií) do nezáporných celých čísel.

Minimalizovať znamená hľadať formulu, ktorá reprezentuje zadanú Booleovskú funkciu a má zo všetkých formulí pre danú funkciu tú najmenšiu mieru. Ako rozhodovací problém môžeme definovať minimalizáciu nasledovne:

Minimalizácia vzhľadom k miere μ.

Inštancia: Funkcia f a číslo k.

Otázka: Existuje formula φ reprezentujúca f s mierou najviac k?

Pre Booleovskú formulu môže byť mierou minimality napríklad hĺbka formuly (akú výšku bude mať syntaktický strom pre danú formulu). V prípade, že spojky AND a OR používame ako viacárne, DNF a CNF majú hĺbku dva, čo je najmenšia možná. Iná možná miera je počet spojok použitých pri zápise formuly.

Najčastejší spôsob zápisu funkcie f na vstupe je jej DNF resp. CNF zápis a hľadaná formula má byť tiež v DNF resp. CNF. V tomto prípade sú dve najčastejšie používané miery počet termov/klauzúl a počet výskytov literálov. Počet výskytov literálov je súčet dĺžok všetkých termov/klauzúl. Počet klauzúl je vhodnou mierou ak nám CNF reprezentuje podmienky, ktoré chceme splniť. Pri menšej CNF je podmienok, ktoré potrebujeme splniť, menej, čo nám môže urýchliť nájdenie riešenia splňujúceho všetky podmienky (toto riešenie bude samozrejme spĺňať aj pôvodné podmienky). [1]

Problémy minimalizácie CNF a DNF by sa zapísali takto:

Minimalizácia CNF vzhľadom k miere μ (kde μ je buď počet klauzúl alebo počet výskytov literálov)

Inštancia: CNF φ a číslo k.

Otázka: Existuje CNF φ' ekvivalentná s φ, ktorá má mieru $\mu(\varphi')$ najviac k?

[1] V oboch prípadoch ide o maskovanú mieru „počet spojok AND a OR“. Pri minimalizácii počtu literálov považujeme spojky za binárne, pri minimalizácii termov za viacárne.

Minimalizácia DNF vzhľadom k miere μ (kde μ je buď počet termov alebo počet výskytov literálov

Inštancia: DNF φ a číslo k.

Otázka: Existuje DNF φ' ekvivalentná s φ, ktorá má mieru $\mu(\varphi')$ najviac k?

Problémy vyzerajú prakticky rovnako a tak nie je prekvapivým nasledujúce zistenie.

Veta 4.1.1. Minimalizácia CNF je zložitosťou ekvivalentná minimalizácii DNF.

Dôkaz. Prevod je oboma smermi identický. Ukážeme si prevod z minimalizácie CNF na minimalizáciu DNF.

Nech (φ, k) je vstupná inštancia problému minimalizácia CNF. Vyrobíme DNF ψ, ktorá bude reprezentovať funkciu duálnu k funkcii reprezentovanej formulou φ tak, že vo φ nahradíme každé AND za OR a každé OR za AND. Inštancia problému minimalizácia DNF bude (ψ, k). Ak (φ, k) je pozitívna inštancia problému minimalizácia CNF, existuje CNF φ' ekvivalentná s φ, ktorá má mieru najviac k. Pre (ψ, k) potom formula duálna k φ' bude hľadanou DNF s mierou najviac k a preto (ψ, k) bude pozitívna inštancia minimalizácie DNF. Opačne ak (ψ, k) je pozitívna inštancia minimalizácie DNF, existuje ψ' dlhé najviac k a formula duálna k ψ' bude hľadaná φ'. Ukázali sme, že inštancia minimalizácie CNF je pozitívna práve vtedy keď je pozitívna inštancia minimalizácie DNF, čím je prevod ukončený.

Prevod z minimalizácie DNF na minimalizáciu CNF je prakticky identický. □

4.2 Veľkosti niektorých funkcií

Zaujímavá je otázka, ako veľké sú najmenšie možné DNF alebo CNF reprezentácie funkcií. Najmenšie minimálne reprezentácie existujú zrejme pre konštantnú funkciu. Formálne je to v prípade DNF jeden prázdny term pre jedničku a prázdna DNF (nula termov) pre nulu. V prípade CNF by to bola jedna prázdna klauzula pre nulu a prázdna CNF pre jedničku.

Pre najväčšiu minimálnu reprezentáciu Booleovskej funkcie v DNF alebo CNF nám poslúži parita. Parita je funkcia, ktorá vráti jedničku práve vtedy, keď je na vstupe nepárny počet jedničiek.

Pozrime sa na DNF pre paritu. Vytvoríme DNF z mintermov podľa pozorovania 2.1.1. Na n premenných má parita $2^n/2$ truepointov, takže v DNF bude $2^n/2$ termov. Všetky tieto termy sú ale primárne implikanty. Všetky primárne implikanty parity obsahujú všetky premenné. Ak by niektorý neobsahoval premennú x, pokrýval by dve ohodnotenia, ktoré sa líšia iba v jednom bite (premennej x) a parita z týchto ohodnotení pokrýva iba jedno. Všetky tieto implikanty sú mintermy, každý pokrýva iba jeden truepoint parity a žiadne dva nie sú identické a preto nie je možné žiaden odobrať bez zmeny funkcie. Parita má iba jednu reprezentáciu v DNF a tá má exponenciálne mnoho primárnych implikantov.

Rovnakým spôsobom zistíme, že najmenšia možná CNF pre paritu je tiež exponenciálne veľká k počtu premenných.

Existujú ale aj funkcie, ktoré síce nie sú exponenciálne dlhé, ale majú exponenciálny počet primárnych implikantov. Príkladom takejto funkcie je $x_1x_2x_3 \ldots x_n + \overline{x_1}y_1 + \overline{x_2}y_2 + \ldots + \overline{x_n}y_n$. Pomocou konsenzov vieme vyrobiť každý z implikantov $\alpha_1\alpha_2 \ldots \alpha_n$, kde každá α je buď x alebo y (nezávisle na ostatných α), pričom žiaden z týchto implikantov nie je absorbovaný iným, takže všetky sú primárne. Primárnych implikantov je teda $2^n + n$ na $2n$ premenných. Ekvivalentnú funkciu by sme našli pre CNF (kde by bol exponenciálny počet implikátov).

Existujú aj funkcie, ktoré sú v DNF krátke a v CNF dlhé (a iné, u ktorých je to opačne). Príkladom takej funkcie je $\bigvee_{i=1}^{n}(x_i \wedge y_i)$. Po roznásobení dostaneme CNF, v ktorej každá klauzula je tvaru $\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n$, kde každá α je buď x alebo y (nezávislé na ostatných). Žiaden implikát neabsorbuje iný (všetky sú rovnako dlhé) a žiaden sa nedá pridať pomocou konsenzov, pretože nikde nie je žiaden negatívny literál. V tejto funkcii je pri $2n$ premenných počet termov v DNF n, ale počet klauzúl v CNF je 2^n. Pre funkciu, ktorá je krátka v CNF, ale dlhá v DNF stačí vziať funkciu duálnu k tejto.

4.3 Ekvivalencia formulí

Pri riešeni minimalizácie zisťujeme, či pre vstup existuje ekvivalentná formula.

Ekvivalencia formulí

Inštancia: Formula φ a formula ψ.

Otázka: Sú φ a ψ ekvivalentné (reprezentujú tú istú funkciu)?

Veta 4.3.1. Problém „Ekvivalencia formulí“ je coNP úplný.

Dôkaz. Problém zrejme patrí do coNP. Certifikátom pre zápornú odpoveď je ohodnotenie premenných, na ktorých sa φ a ψ líšia.

coNP ťažkosť dokážeme prevodom problému splniteľnosti na negáciu problému „ekvivalencia formulí“. Majme χ inštanciu problému splniteľnosti. Potom $(\chi, 0)$ je inštancia negácie problému „Ekvivalencia formulí“. Ak je χ pozitívna inštancia, tak χ a 0 nie sú ekvivalentné. Ak je χ negatívna inštancia, tak χ a 0 sú ekvivalentné. □

Poznámka 4.3.2. Pokiaľ sú formuly v CNF alebo v DNF, problém zostáva coNP úplný. Pri CNF ostáva dôkaz nezmenený, pri DNF budeme porovnávať χ s jedničkou.

Kapitola 5

Minimalizácia DNF – Umansov dôkaz

5.1 Σ_2 a Shortest Implicant Core

Christopher Matthew Umans vo svojej dizertačnej práci [13] z roku 1996 dokázal, že minimalizácia DNF formulí je Σ_2 úplná pri minimalizácii počtu literálov aj pri minimalizácii počtu termov.

Zopakujme si, že pre triedu Σ_2 je známy Σ_2 úplný problém zvaný QSAT_2. Ten ostáva Σ_2 úplný aj v prípade obmedzenia φ na DNF [7, Theorem 4]. V nasledujúcom texte budeme ako QSAT_2 označovať práve tento variant problému.

QSAT_2

Inštancia: Kvanitfikovaná Booleovská formula tvaru $\exists X_1 \forall X_2 \varphi(X_1, X_2)$ kde X_1 a X_2 sú disjunktným rozkladom množiny premenných vo formule φ a φ je v DNF.

Otázka: Je hodnota φ rovná jednej?

Pri dôkaze Σ_2 úplností minimalizácií DNF budeme používať iný Σ_2 úplný problém nazvaný „Shortest Implicant Core“[1] a to dokonca dva jeho varianty.

[1]Možný preklad by bol „najkratšie jadro implikantu“. Tento preklad je však trochu kostrbatý a tak ostaneme pri pôvodnom anglickom názve.

Shortest Implicant Core 1 - SIC1

Inštancia: DNF formula $\varphi = t_1 \vee t_2 \vee \ldots \vee t_l$ a celé číslo k.

Otázka: Existuje implikant C formule φ, ktorý absorbuje t_l a je kratší ako k?

Shortest Implicant Core 2 - SIC2

Inštancia: DNF formula φ, jej implikant C a celé číslo k.

Otázka: Existuje implikant C' formule φ, ktorý absorbuje C a je kratší ako k?

Veta 5.1.1. Problém SIC1 je ekvivalentný problému SIC2.

Dôkaz. Prevedieme najprv SIC1 na SIC2. Ak $(\varphi = t_1 \vee t_2 \vee \ldots \vee t_l, k)$ je inštancia problému SIC1, inštanciou problému SIC2 bude (φ, t_l, k). Je zrejmé, že buď sú obe inštancie pozitívne alebo obe negatívne, pretože v oboch sa hľadá implikant tej istej funkcie, ktorý absorbuje t_l.

Prevod z SIC2 na SIC1 je podobný. Pre inštanciu (φ, C, k) problému SIC2 vyrobíme inštanciu $(\varphi \vee C, k)$. Vďaka leme 2.2.8 vieme, že $\varphi \vee C$ reprezentuje tú istú funkciu ako φ, a preto v oboch prípadoch hľadáme implikant tej istej funkcie, ktorý absorbuje C. Inštancie sú preto buď obe pozitívne alebo obe negatívne.

Oba problémy sú teda ekvivalentné. □

Veta 5.1.2. Problém „Shortest Implicant Core 1“ je Σ_2 úplný.

Dôkaz. Problém zrejme patrí do Σ_2. Certifikátom bude hľadaný implikant C, overíme, že spĺňa zadanie (absorbuje t_l a je kratší ako k) a pomocou coNP orákula overíme, že je skutočne implikantom φ. Po aplikovaní čiastočného ohodnotenia reprezentovaného termom C musí byť φ tautológiou. Zistenie, či je výsledná formula tautológiou je riešiteľné pomocou coNP orákula.

Pre dôkaz Σ_2 ťažkosti použijeme dôkaz z [11].

Majme inštanciu $< X_1, X_2, \varphi >$ problému QSAT_2. Bližší pohľad na problém QSAT_2 odhalí, že hľadáme čiastočné ohodnotenie A pre premenné z X_1

také, pre ktoré je $\varphi[A]$ tautológiou. Keďže čiastočné ohodnotenie je reprezentovateľné implikantom a problém, na ktorý prevádzame je hľadanie implikantu, budeme hľadať taký implikant, ktorý kóduje ohodnotenie premenných z X_1.

Označme $m = |X_1|$ a nech X_1 obsahuje premenné $x_1, x_2, \ldots x_m$. Vyrobíme si $2m$ nových premenných $a_1, a_2, \ldots a_m$, a $b_1, b_2, \ldots b_m$. Nahradíme každý výskyt literálu x_i v φ premennou a_i a každý výskyt literálu $\overline{x_i}$ premennou b_i a vzniknutú formulu označíme φ'.

Lema 5.1.3. Inštancia QSAT_2 je pozitívna práve keď existuje implikant formuly φ', ktorý obsahuje práve jeden literál z každého páru (a_i, b_i).

Dôkaz. ($\Leftarrow$) To, že je inštancia problému QSAT_2 pozitívna znamená, že je pravdivá formula $\exists X_1 \forall X_2 \varphi$, teda existuje ohodnotenie premenných z X_1, ktoré zaručí pravdivosť φ. Tomuto ohodnoteniu zodpovedá implikant na premenných z X_1. V tomto implikante je práve jeden z každej dvojice literálov $(x_i, \overline{x_i})$ a nahradením x_i za a_i a $\overline{x_i}$ za b_i vznikne implikant formuly φ', ktorý obsahuje práve jeden literál z každého páru (a_i, b_i).

($\Rightarrow$)Ak máme implikant I pre φ', ktorý obsahuje práve jeden literál z každého páru (a_i, b_i), vyrobíme priamočiaro ohodnotenie premenných x_i tak, že pre i, kde je v implikante vybraný literál a_i nastavíme hodnotu jedna a pre i, kde je vybraný literál b_i nastavíme hodnotu nula. φ s týmto ohodnotením je zrejme totožná s φ' s ohodnotaním definovaným implikantom I a teda je rovná konštantnej jedničke. [2] □

Už je možné vidieť, že budeme hľadať implikant obsahujúci práve jeden literál z každej dvojice (a_i, b_i), ktorý bude mať dĺžku m a bude absorbovať implikant $t_l = a_1 b_1 a_2 b_2 \ldots a_m b_m$. Voľbou absorbovaného implikantu máme zaručené, že hľadaný implikant bude obsahovať iba premenné a_i a b_i. Potrebujeme ešte zaručiť, že každá tam bude práve raz.

[2] Umans tu používa dôkaz nepriamo (ak QSAT_2 nie je pravdivý tak implikant neexistuje).

Zadefinujeme si formulu f, kde z_i sú nové premenné, nasledovne:

$$\begin{aligned}
f = \quad & a_1 z_1 \vee & & b_1 z_1 \\
\vee \quad & \overline{z_1} a_2 z_2 \vee & & \overline{z_1} b_2 z_2 \\
\vee \quad & \overline{z_2} a_3 z_3 \vee & & \overline{z_2} b_3 z_3 \\
& \vdots & & \vdots \\
\vee \quad & \overline{z_{m-2}} a_{m-1} z_{m-1} \vee & & \overline{z_{m-2}} b_{m-1} z_{m-1} \\
\vee \quad & \overline{z_{m-1}} a_m \vee & & \overline{z_{m-1}} b_m
\end{aligned}$$

Lema 5.1.4. Nech C je konjunkcia, ktorá obsahuje iba literály z množiny $\{a_1, b_1, a_2, b_2, \ldots, a_m, b_m\}$. C je implikantom f práve vtedy keď C obsahuje *aspoň jeden* literál z každého páru (a_i, b_i).

Dôkaz. ($\Rightarrow$)Predpokladajme, že C z páru (a_i, b_i) vynechalo oba literály. Zostrojíme ohodnotenie A pokryté implikantom C, ktoré ale nesplní f. Nastavíme premenné v C tak, aby C bolo splnené. Pre $j \geq i$ nastavíme z_j na jedničku a všetky ostatné premenné, ktoré ešte nie sú nastavené nastavíme na nulu. Toto ohodnotenie spĺňa C ale nespĺňa f.

($\Leftarrow$)Ak C obsahuje aspoň jeden literál z každého páru (a_i, b_i), vyberieme si z neho term, ktorý bude z každého páru (a_i, b_i) obsahovať práve jeden literál. Tento term bude absorbovať C a zároveň bude implikantom f, pretože je možné zostrojiť ho konsenzuálnou metódou použitou na f. □

Vytvoríme si formulu $\varphi'' = (f \wedge \varphi') \vee a_1 b_1 a_2 b_2 \ldots a_m b_m$. Ak prvú časť roznásobíme do DNF, stále zostane polynomiálne veľká. (φ'', m) bude vstupom pre problém „Shortest Implicant Core“, kde t_l bude práve $a_1 b_1 a_2 b_2 \ldots a_m b_m$.

Lema 5.1.5. Formula φ'' má implikant C absorbujúci t_l dlhý najviac m práve vtedy keď inštancia QSAT_2 je pozitívna.

Dôkaz. ($\Rightarrow$)Predpokladajme, že máme taký implikant C, ktorý spĺňa zadanie. Aby C bol implikant φ'', musí byť buď implikantom $(f \wedge \varphi')$ alebo implikantom t_l. Keďže absorbuje t_l a je kratší ako t_l, nemôže byť implikantom t_l, preto musí byť implikantom $(f \wedge \varphi')$. Keďže je implikantom $(f \wedge \varphi')$, musí byť implikantom ako f, tak aj φ'. Podľa lemy 5.1.4 musí z každého páru (a_i, b_i) obsahovať aspoň jeden literál, aby bol implikantom f. C je ale dlhé najviac m a preto z každého páru (a_i, b_i) práve jeden literál. C je implikantom φ' a obsahuje z každého páru (a_i, b_i) práve jeden literál a tak podľa lemy 5.1.3 je inštancia QSAT_2 pozitívna.

($\Leftarrow$) Podľa lemy 5.1.3, ak je inštancia QSAT_2 pozitívna, existuje implikant formuly φ' obsahujúci práve jeden literál z každého páru (a_i, b_i). Podľa lemy 5.1.4 je takýto implikant zároveň implikantom formuly f a teda je aj implikantom pre φ''. □

Ukázali sme, že inštancia (φ'', m) problému „Shortest Implicant Core 1" je pozitívna práve keď je pozitívna inštancia (X_1, X_2, φ) problému QSAT_2. Problém „Shortest Implicant Core 1" je teda Σ_2 ťažký a zároveň do Σ_2 patrí, je teda Σ_2 úplný. □

Dôsledok 5.1.6. Problém „Shortest Implicant Core 2" (SIC2) je Σ_2 úplný.

Dôkaz. Podľa lemy 5.1.1 je problém SIC2 ekvivalentný problému SIC1 a SIC1 je podľa vety 5.1.2 Σ_2 úplný. □

5.2 Minimalizácia výskytov literálov

Teraz môžeme pristúpiť k dôkazu Σ_2 úplnosti minimalizácie Booleovských formulí.

Minimalizácia počtu literálov v DNF

Inštancia: DNF φ a celé číslo k.

Otázka: Existuje DNF ψ ekvivalentná φ obsahujúca najviac k literálov?

Veta 5.2.1. Problém „Minimalizácia počtu literálov v DNF" je $\Sigma_2$úplný.

Dôkaz. Problém je zrejme v Σ_2. Ako certifikát uhádneme DNF obsahujúcu najviac k výskytov literálov a pomocou coNP orákula overíme, že je na všetkých ohodnoteniách premenných zhodná s φ.

Σ_2 ťažkosť sa ukáže prevodom z problému „Shortest Implicant Core 1". Použitý dôkaz je z [11].

Nech $(\varphi = t_1 \vee t_2 \vee \ldots \vee t_l, k)$ je inštancia problému „Shortest Implicant Core 1". Vyrobíme takú inštanciu problému „Minimalizácia počtu literálov v DNF", v ktorej vynútime zachovanie všetkých termov okrem posledného v nezmenenej podobe v každej kratšej DNF reprezentujúcej funkciu reprezentovanú φ.

Vyrobíme φ' roznásobením $\varphi \wedge \neg t_l$ do DNF. Tým zaručíme, že φ' nebude pokrývať žiadne ohodnotenia pokryté t_l, takže keď t_l pridáme, nebude redundantným implikantom.

Označme $s_1, s_2, \ldots s_m$ jednotlivé termy φ' roznásobeného do DNF a $m' = \max(m, k+1)$. Zadefinujeme

$$\varphi'' = t_l w_1 w_2 w_3 \ldots w_{m'} \vee \bigvee_{i=1}^{m} s'_i$$

kde $s'_i = s_i w_1 w_2 \ldots w_{i-1} w_{i+1} \ldots w_{m'}$. Tiež označíme $k' = k + m' + \sum_{i=1}^{m} |s'_i|$. Inštancia problému„Minimalizácia počtu literálov v DNF“ bude (φ'', k').

Lema 5.2.2. Ak C je implikant φ absorbujúci t_l, tak $\varphi'' = C w_1 w_2 \ldots w_m \vee \bigvee_{i=1}^{m} s'_i$.

Dôkaz. Už vieme, že ak nahradíme implikant v DNF iným implikantom, ktorý ho absrbuje, DNF sa nezmení (lema 2.2.10). Stačí teda ukázať, že $C w_1 w_2 \ldots w_m$ je implikantom φ''. Všimneme si, ako sme vytvárali φ''. Keď za všetky w_i dosadíme jedničky, dostaneme $\varphi' \vee t_l$, čo je ale ekvivalentné φ. Keďže C je implikantom φ, je $C w_1 w_2 \ldots w_m$ (nastavujúci všetky w_i na jedničky) implikantom φ''. □

Ak má implikant C dĺžku menšiu alebo rovnú k, tak $C w_1 w_2 \ldots w_m \vee \bigvee_{i=1}^{m} s'_i$ má počet literálov najviac k'. Ak teda máme pozitívnu inštanciu problému „Shortest Implicant Core 1“, máme aj pozitívnu inštanciu problému „Minimalizácia počtu literálov v DNF“. Potrebujeme ešte ukázať opačnú implikáciu.

Lema 5.2.3. Nech f je DNF ekvivalentná φ''. Potom pre všetky i platí a) f obsahuje term s''_i taký, že s'_i absorbuje s''_i, b) celková dĺžka týchto termov je aspoň $\sum_{i=1}^{m} |s'_i|$ a c) žiaden z týchto termov nepokrýva ľubovoľné ohodnotenie pokryté termom $t_l w_1 w_2 \ldots w_{m'}$.

Dôkaz. Nech A_i je ohodnotenie spĺňajúce s'_i s w_i nastaveným na nulu a nech s''_i je term z f, ktorý toto ohodnotenie pokrýva. Ak s''_i vynecháva ľubovoľné w_j pre $j \neq i$, pokrýva tak ohodnotenie, v ktorom sú w_i aj w_j nastavené na nulu. Takéto ohodnotenie ale f nepokrýva. Ak s''_i vynechá nejaký literál z s_i, tak pokrýva nejaké ohodnotenie nekonzistentné s s'_i, ale s w_i nastaveným na nulu. To ale nejde, pretože ohodnotenia, ktoré spĺňajú φ'' a majú nastavené w_i na nulu spĺňajú s_i. s'_i musí absorbovať s''_i, teda $|s'_i| \leq |s''_i|$. Keďže s''_i obsahuje všetky w_j pre $j \neq i$, nemôže pokrývať iné ohodnotenie A_j (pre

$j \neq i$). A_i sú ale navzájom rôzne, preto sú rôzne aj s_i'' a ich celková dĺžka musí byť aspoň $\sum_{i=1}^{m} |s_i'|$. A nakoniec žiadne z s_i'' nepokrýva ohodnotenia pokryté $t_l w_1 w_2 \dots w_{m'}$, pretože obsahujú (sú absorbované) termy s_i' a tie sme tvorili tak, aby nepokrývali ohodnotenia pokryté t_l. □

Okrem termov s_i'' musí DNF ekvivalentné s φ'' obsahovať aj termy, ktoré pokryjú ohodnotenia pokryté termom $t_l w_1 w_2 \dots w_{m'}$. Ak má mať ekvivalentná DNF dĺžku najviac k', tieto ostatné termy môžu mať dĺžku najviac $k' - \sum_{i=1}^{m} |s_i'| \leq k + m'$. Ľubovoľný implikant, ktorý pokrýva niektoré z ohodnotení pokryté $t_l w_1 w_2 \dots w_{m'}$ musí obsahovať všetky literály $w_1 w_2 \dots w_{m'}$, pretože ináč by pokrýval ohodnotenie konzistentné s t_l, ale s niektorým w_i nastaveným na nulu a φ'' takéto ohodnotenie nepokrýva. Preto každý ďalší term musí mať dĺžku aspoň m'. Keďže ale $k + m' < 2m'$ (spomeňme si, že $m' = \max(m, k+1)$), takýto term tam môže byť najviac jeden. Tento term označíme C. C má dĺžku najviac $k + m'$, absorbuje $t_l w_1 w_2 \dots w_{m'}$ a obsahuje každé w_i. C je implikantom φ'', ale φ'' s dosadením jedničiek za každé w_i je ekvivalentné φ, takže $C' = C \setminus w_1 w_2 \dots w_{m'}$, čo je ekvivalentné C so všetkými w_i nastavenými na jedničku je implikantom φ s dĺžkou najviac k, ktorý absorbuje t_l.

Ukázali sme, že inštancia (φ, k) je pozitívnou inštanciou problému „Shortest Implicant Core 1“ práve vtedy, keď (φ'', k) je pozitívnou inštanciou problému „Minimalizácia počtu literálov v DNF“. Problém je Σ_2 úplný. □

5.3 Minimalizácia počtu termov

Minimalizácia počtu termov v DNF

Inštancia: DNF φ a celé číslo k.

Otázka: Existuje DNF ψ ekvivalentná φ obsahujúca najviac k termov?

Veta 5.3.1. Problém „Minimalizácia počtu termov v DNF“ je Σ_2 úplný.

Dôkaz. To, že problém patrí do Σ_2 sa ukáže rovnako ako v prípade predchádzajúceho problému. Certifikátom je formula s menej ako k termami a jej ekvivalencia s φ sa overí coNP orákulom.

Σ_2 ťažkosť sa ukáže prevodom z problému „Shortest Implicant Core 2“ (SIC2). Inštanciou problému SIC2 je formula φ, jej implikant C a celé číslo k.

Prvý dôkaz tvrdenia sa dá nájsť v [13], kde je jedno z pozorovaní nedostatočne zdôvodnené. Upravený a zjednodušený dôkaz sa dá nájsť v [12]. Tu použijeme upravenú kombináciu oboch dôkazov.

Nech (φ, C, k) je inštanciou problému „Shortest Implicant Core 2“, kde $C = x_1 x_2 \dots x_n$ a $\varphi = t_1 \vee t_2 \vee \dots \vee t_l$. Bez ujmy na všeobecnosti môžeme predpokladať, že všetky premenné v C sú pozitívne.

Roznásobme $\varphi \wedge \neg C$ do DNF a termy tejto DNF si označme $s_1, s_2, \dots, s_m$. Z toho, ako vznikli, vyplýva, že každý z nich obsahuje aspoň jeden z literálov $\overline{x_j}, 1 \leq j \leq n$. Ďalej ako p_i označme i-ty term jednoznačnej DNF reprezentácie parity na nových premenných $a_1, a_2, \dots a_r$, kde ako r zvolíme $m \times n$. [3]

Inštancia problému „Minimalizácia počtu termov v DNF“ bude

$$\varphi' = \bigvee_{i=1}^{m} s'_i \vee \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} u_{i,j}$$

kde $s'_i = s_i z_1 z_2 \dots z_{i-1} z_{i+1} \dots z_m$ a $u_{i,j} = (p_i)\overline{x_j} z_1 z_2 \dots z_m$, a číslo $k' = m(k+1)$.

Prvú časť φ' si označíme ako $T = \bigvee_{i=1}^{m} s'_i$. Tiež si zadefinujeme funkciu $P(a_1, a_2, \dots a_r) = \bigvee_{i=1}^{m} p_i$, aby sme sa na ňu vedeli odkazovať.

Pozorovanie 5.3.2. Nech A je dosadenie, ktoré nastavuje všetky premenné z na jedničku a aspoň jednu z premenných x na nulu. Potom $\varphi(A) = T(A)$.

Dôkaz. Nech R je čiastočné dosadenie, ktoré všetkým premenným z nastaví jedničku. V tom prípade $T[R] \equiv \bigvee_{i=1}^{m} s_i \equiv \varphi \wedge \neg C \equiv \varphi \wedge \bigvee_{j=1}^{n} \overline{x_j}$. Z toho je tvrdenie zrejmé. □

Lema 5.3.3. Ak existuje implikant C' funkcie φ, ktorý absorbuje C, veľkosti najviac k, tak φ' má ekvivalentnú DNF φ'' s najviac $m(k+1)$ termami.

[3] Z týchto termov budeme potrebovať iba prvých m, preto nevadí, že je príslušná DNF parity exponenciálne veľká. V pôvodnom dôkaze bol počet premenných a zvolený ako $\log_2 2m$, čo je najmenší počet, aby bol termov dostatok. V [12] sa už nepoužíva parita, ale miesto každého termu p_i je použitá premenná a_i. Naša voľba sa však využije v nasledujúcej kapitole.

Dôkaz. Ak máme C' daných vlastností, zostrojíme φ'' a ukážeme, že je ekvivalentná s φ'.

$$\varphi'' = \bigvee_{i=1}^{m} s'_i \vee \bigvee_{i=1}^{m} \bigvee_{x_j \in C'} u_{i,j}$$

Keďže φ'' obsahuje podmnožinu termov φ', zrejme $\varphi'' \implies \varphi'$. Potrebujeme ukázať obrátenú implikáciu. Nech A je ohodnotenie, pre ktoré je φ' rovné jednej. Keďže φ' nepokrýva žiadne ohodnotenia so všetkými premennými x nastavenými na jedničku (každý term obsahuje niektoré x negované), aspoň jedna z premenných x je v A nastavená na nulu. Ak A nastavuje aspoň jednu z premenných z na nulu alebo ak $P(A) = 0$, A musí byť pokryté funkciou T, pretože ani jeden z termov $u_{i,j}$ takéto ohodnotenie nepokrýva, a keďže φ'' je disjunkcia T a ďalších termov, tak $\varphi''(A) = 1$. Ináč A nastavuje všetky premenné z na jedničku a $P(A) = 1$. Rozoberieme dve možnosti podľa toho, či je A konzistentné s C'.

Ak A je konzistentné s C', čiže všetky $x_j \in C'$ sú nastavené na jedničku, implikuje to $\varphi(A) = 1$ (C' je implikantom φ) a podľa pozorovania 5.3.2 $T(A) = 1$, čiže aj $\varphi''(A) = 1$. Ak A nie je konzistentné s C', nastavuje niektorú z premenných $x_j \in C'$ na nulu a preto je vo φ'' pokryté niektorým z termov $u_{i,j} = (p_i)\overline{x_j}z_1z_2 \dots z_s$.

Vo všetkých prípadoch $\varphi''(A) = 1$ ak $\varphi'(A) = 1$. $\varphi'' \equiv \varphi'$. Formula φ'' má $m + mk = m(k + 1)$ termov, ako požaduje znenie lemy. □

Lema 5.3.4. Ak má φ' ekvivalentnú DNF φ'' s najviac $m(k + 1)$ termami, existuje implikant C' funkcie reprezentovanej φ absorbujúci C dlhý najviac k.

Dôkaz. Nech φ'' je DNF ekvivalentná φ' obsahujúca najviac $m(k+1)$ termov. Cieľom je zostrojiť pomocou nej implikant C' pre φ, absorbujúci C, dĺžky najviac k.

Rozdelíme termy φ'' do dvoch množín: S_1 a S_2 tak, že S_1 bude obsahovať tie termy, ktoré obsahujú kompletný niektorý z termov p_i a S_2 všetky ostatné termy. O termoch z S_2 učiníme dve pozorovania.

Pozorovanie 5.3.5. Každý term t z S_2 implikuje T. Predpokladajme, že nie. V tom prípade bude existovať ohodnotenie konzistentné s t pre ktoré $T(A) = 0$. Pretože t neobsahuje ani jeden z p_i úplný, môžeme modifikovať ohodnotenie premenných a v A tak, aby P bolo v tomto modifikovanom ohodnotení A' nepravdivé, ale t ostalo pravdivé. [4] Keďže T na premenných

[4] Modifikujeme tie premenné a, ktoré v t nie sú.

a nezávisí, $T(A') = 0$. Lenže $P(A') = 0$ a $T(A') = 0$ znamená $\varphi''(A') = 0$, čo je nekonzistentné s tým, že t je implikantom φ''.

Pozorovanie 5.3.6. V S_2 je aspoň m navzájom rôznych termov. [5] Pre každý term s'_i nech A_i je ohodnotenie konzistentné s s'_i, ktoré nastavuje z_i na nulu a premenné a tak, že $P(A) = 0$. Každé z ohodnotení A_i musí byť pokryté nejakým termom s''_i z φ''. Keďže pre všetky A_i je $P(A_i) = 0$, všetky tieto termy musia patriť do S_2. Term s''_i nemôže obsahovať literál z_i, pretože ohodnotenie A_i, ktoré pokrýva, má premennú z_i nastavenú na nulu. Term s''_i ale musí obsahovať literál z_l pre každé $l \neq i$, pretože ináč by pokrýval ohodnotenie s dvoma premennými z nastavenými na nulu, a také ohodnotenie φ'' nepokrýva. Každý z termov s''_i teda vynecháva práve jednu z premenných z a každý inú. Termy s''_i sú preto navzájom rôzne, je ich m a všetky sú v S_2.

Zameriame sa teraz na termy z S_1. Každý z týchto termov obsahuje niektorý z termov p_i pre $1 \leq i \leq m$ neskrátený. Ďalej každý z termov z S_1 obsahuje negovanú niektorú z premenných x, pretože φ'' nepokrýva ohodnotenie, ktoré by malo všetky x nastavené na jedničku. Každý z termov $t \in S_1$ môžeme označiť dvojicou (i, j) tak, že t obsahuje term p_i a literál $\overline{x_j}$. Pri viacerých možnostiach označenia si jednu ľubovoľne vyberieme.

Podľa Dirichletovho princípu musí existovať i_0 pre ktoré existuje najviac $|S_1|/m$ termov označkovaných $(i_0, *)$. Do C' zahrnieme všetky literály x_j, pre ktoré existuje term so značkou (i_0, j).

Ukážeme, že C' je implikantom φ. Predpokladajme, že nie je. Existuje teda ohodnotenie A konzistentné s C', v ktorom je φ nulové. Keďže $C = x_1 x_2 \dots x_n$ je implikantom φ, toto ohodnotenie A musí niektorú z premenných x nastaviť na nulu. Upravíme ohodnotenie A tak, aby všetky premenné z boli nastavené na jedničku a premenné a boli nastavené tak, že spĺňajú p_{i_0}. Tieto premenné nie sú ani vo φ ani v C', takže hodnota φ ani C' sa touto úpravou nezmení. Ukážeme, že v tom prípade $\varphi'(A) = 1$, ale $\varphi''(A) = 0$, teda $\varphi' \not\equiv \varphi''$.

Tým, ako sme vytvorili A platí $\varphi'(A) = 1$, pretože φ' pokrýva všetky ohodnotenia konzistentné s niektorým z termov p_i, niektorým x nastaveným na nulu a so všetkými z nastavenými na jedničku. Rovnako ale $\varphi(A) = 0$, a teda podľa lemy 5.3.3 platí $T(A) = 0$, čo znamená, že A falzifikuje každý term v S_2 podľa pozorovania 5.3.5. Navyše A falzifikuje aj každý term t v S_1 pretože buď nastavuje na jedničku niektorú z premenných x, ktorá je v

[5] Toto je pozorovanie, ktoré je v dôkaze v [13] nejasné.

terme t negovaná, alebo term t obsahuje p_i rôzny od p_{i_0} a spomedzi termov p je A konzistentné iba s p_{i_0}. Keďže φ'' pozostáva iba z termov z S_1 a S_2, $\varphi''(A) = 0$, čo je spor s $\varphi' \equiv \varphi''$. Preto C' musí byť implikantom φ.

Nakoniec overíme, že má správnu dĺžku:

$$|C'| \leq \frac{|S_1|}{s'} \leq \frac{s'(k+1) - s'}{s'} = k$$

□

Lemy 5.3.3 a 5.3.4 ukazujú, že inštancia (φ, C, k) problému „Shortest Implicant Core 2“ je pozitívna práve vtedy, keď je pozitívna inštancia $(\varphi', m(k+1))$ problému „Minimalizácia počtu termov v DNF“, čím je ukázaná Σ_2 ťažkosť problému. Problém „Minimalizácia počtu termov v DNF“ je teda Σ_2 úplný. □

Kapitola 6

Matched formuly

6.1 Základné pojmy a vlastnosti

Matched formuly sú jednou z tried CNF, pre ktoré je problém splniteľnosti triviálny – splnené sú vždy. Na ich charakterizovanie budeme potrebovať nasledujúci pojem.

Definícia (Incidenčný graf)**.** Incidenčným grafom CNF nazveme neorientovaný bipartitný graf, ktorého vrcholmi v jednej partite je množina premenných a v druhej partite množina klauzúl. Hrana medzi premennou x a klauzulou C vedie práve vtedy, keď C obsahuje literál x alebo $\neg x$.

Pozorovanie 6.1.1. Incidenčný graf CNF nezávisí na negácii jednotlivých premenných v klauzulách. Viacero CNF môže mať rovnaký graf.

Poznámka 6.1.2. Incidenčný graf môžeme analogicky zadefinovať pre DNF.

Definícia (Matched CNF)**.** Matched CNF je taká CNF, ktorej graf má párovanie pokrývajúce všetky klauzule.

Poznámka 6.1.3. Podobným spôsobom môžeme zadefinovať matched DNF.

Matched formuly boli definované v [2], odkiaľ pochádza aj nasledujúce pozorovanie.

Pozorovanie 6.1.4. Každá Matched CNF formula je splniteľná.

Dôkaz. Párovanie v grafe matched CNF priradí každej klauzule práve jednu premennú, ktorej ohodnotenie nastavíme tak, aby bola daná klauzula splnená. Týmto splníme všetky klauzuly. □

Poznámka 6.1.5. Pre matched DNF je z tých istých dôvodov jednoduchý problém „FALS“.

Pozorovanie 6.1.6. Trieda matched formúl je uzavretá na negácie literálov a negácie premenných. Rovnako je uzavretá na odstránenie klauzule (v prípade matched DNF termu). Nie je uzavretá na odstránenie literálu (ako protipríklad môžeme uviesť formulu $(x_1 \vee x_2) \wedge (\neg x_1)$, ktorá je matched, ale po odstránení literálu x_2 už matched nebude).

Pozorovanie 6.1.7. Funkcia s matched formulou môže mať exponeniciálne množstvo primárnych implikátov. Príkladom je formula $(x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_n) \wedge \bigwedge_{i=1}^{n} (\neg x_i \vee y_i)$ s $2n$ premennými. Párovanie priradí každej klauzule s ypsilónom premennú y_i a zvyšnej niektorú z premenných x. Počet primárnych implikátov je ale 2^n.

Formuly z triedy Matched je možné rozpoznať v polynomiálnom čase pomocou algoritmov hľadajúcich párovanie v bipartitnom grafe. Existujú aj zovšeobecnenia matched formúl, kde sa v incidenčnom grafe hľadajú úplné bipartitné podgrafy (pri matched formulách sú to úplné bipartitné grafy na dvoch vrcholoch), prípadne zovšeobecnenie, kde chceme, aby formula bola splniteľná pri ľubovoľnom nahradení literálov ich komplementami (táto trieda sa nazýva Var-SAT a matched formuly do nej patria). Zisťovanie, či formula patrí do niektorého z týchto zovšeobecnení je však ťažké (NP úplné resp. Π_2 úplné) [10].

6.2 Zložitosti u Matched

Matched CNF formuly sú jednou z tried CNF formúl, pre ktoré je problém splniteľnosti jednoduchý. S inými triedami, pre ktoré je tiež splniteľnosť jednoduchá, sú však väčšinou neporovnateľné [14] – výnimku tvorí trieda nazvaná LinAut, ktorá je nadmnožinou matched formúl.

Pre niektoré z týchto tried je problém minimalizácie preskúmaný a je zistené, že je jednoduchší ako pre všeobecné formule (ako príklad môžeme uviesť monotónne formuly, pre ktoré je minimalizácia coNP úplná [4] a takzvané Hornovské formuly, kde je NP úplná [5]), čo by mohlo zvádzať k tomu, že aj pre Matched formuly bude problém minimalizácie jednoduchší ako pre obecné formuly. V nasledujúcej časti ukážeme, že problémy spojené s minimalizáciou sú pre Matched formuly rovnako ťažké ako pre obecné formuly.

Závislosť na premennej

Inštancia: Matched CNF φ a premenná z v nej obsiahnutá.

Otázka: Závisí hodnota φ na ohodnotení premennej z?

Vhodným krokom pri minimalizácii je odstrániť premenné, na ktorých funkcia nezávisí. Ak máme minimálnu CNF/DNF pri niektorej miere a obsahuje premennú, na ktorej nezávisí, odstránením premennej, na ktorej funkcia nezávisí, sa CNF/DNF nepredĺži (premennú odstránime tak, že za ňu dosadíme hodnotu podľa vlastného výberu).

Veta 6.2.1. Pre matched CNF formuly je problém „Závislosť na premennej“ NP úplný.

Dôkaz. To, že je v NP je zrejmé. Certifikátom budú dve ohodnotenia premenných líšiace sa iba v ohodnotení premennej z, na ktorých bude ale rozdielna hodnota φ.

NP ťažkosť ukážeme Prevodom zo splniteľnosti CNF (SAT). Majme CNF ψ, ktorá je inštanciou problému SAT. Z nej vyrobíme inštanciu problému „Závislosť na premennej“, ktorá bude mať kladnú odpoveď práve keď ψ bude splniteľná.

Nech $\psi = C_1 \wedge C_2 \wedge \ldots C_n$. Vyrobíme

$$\varphi = (C_1 \vee w_1) \wedge (C_2 \vee w_2) \wedge \cdots \wedge (C_n \vee w_n) \wedge (w_1 \vee w_2 \vee \cdots \vee w_n \vee z)$$

kde $w_1, w_2, \ldots, w_n, z$ sú nové premenné. φ je zrejme matched. Každej klauzule okrem poslednej priradím príslušnú w_i a poslednej klauzule priradím z.

Tvrdíme: ψ je splniteľná práve keď φ závisí na premennej z.

Ak ψ je splniteľná, tak nech $\vec{x}$ je vektor, na ktorom je splnená. Ohodnoťme všetky premenné w_i nulou. Potom $\varphi[\vec{x}, \vec{w} = \vec{0}, z = 0]$ má hodnotu nula a $\varphi[\vec{x}, \vec{w} = \vec{0}, z = 1]$ má hodnotu jedna. φ teda na z závisí.

Ak je ψ nesplniteľná, tak rozoberieme dva prípady podľa toho, či v ohodnotení premenných vo formule φ existuje nenulové w_i.

Ak nenulové w_i neexistuje, tak $\varphi[\vec{w} = \vec{0}]$ sa zredukuje na $\psi \wedge z$. Keďže ale ψ nie je splniteľné, hodnota tohto výrazu je nulová nezávisle na ohodnotení z.

Ak je niektoré z w_i nenulové, spĺňa tým poslednú klauzulu, ktorá je jedinou klauzulou obsahujúcou z. Hodnota tejto klauzule na z nezáleží a teda na z nezáleží ani hodnota celého φ. □

Pre matched DNF urobíme rovnaký prevod z problému „FALS“.

Pre ďalšie dôkazy budeme potrebovať NP úplnosť obmedzenej verzie tohto problému.

Závislosť na premennej matched CNF s jedným výskytom – ZPM1

Inštancia: Matched formula φ, v ktorej je práve jeden výskyt premennej z.

Otázka: Závisí hodnota φ na ohodnotení premennej z?

Poznámka 6.2.2. Aby sme sa na problém nemuseli odvolávať celým názvom, označíme ho ZPM1.

Lema 6.2.3. Problém „ZPM1“ je NP úplný.

Dôkaz. Použijeme dôkaz vety 6.2.1 bez zmeny. Všimneme si, že formula φ s premennou z tvoria inštanciu problému „ZPM1“. □

Nasledujú ďalšie problémy, ktoré sa ukazujú ťažké pri matched formuliach.

Odstránenie premennej

Inštancia: Matched CNF φ, klauzula $C \in \varphi$, premenná $z \in C$.

Otázka: Je $\psi = \varphi \setminus \{C\} \cup \{C \setminus \{z\}\}$ ekvivalentné φ?

Pre jednoduchosť označme $\varphi \setminus \{C\} \cup \{C \setminus \{z\}\}$ ako $\varphi \setminus z$.

Veta 6.2.4. Problém „Odstránenie premennej“ je coNP úplný.

Dôkaz. Problém zrejme do coNP patrí. Pre zápornú odpoveď je certifikátom ohodnotenie pre ktoré sa φ nerovná ψ.

Prevedieme negáciu problému „ZPM1“ na problém odstránenia premennej. Majme inštanciu (φ, z) problému „ZPM1“, kde φ obsahuje práve jeden výskyt premennej z. Tvrdíme, že premennú z môžeme vyškrtnúť bez zmeny funkcie φ práve vtedy keď φ na z nezávisí. Inštanciou problému „Odstránenie premennej“ bude (φ, C, z), kde C je klauzula φ obsahujúca z.

Nech φ na z závisí (teda sú pozitívnou inštanciou problému ZPM1). V tom prípade $\varphi \setminus z$ nemôže reprezentovať rovnakú funkciu ako φ, pretože $\varphi \setminus z$ už na z nezávisí.

Ak φ na z nezávisí, tak $\varphi[z = 1] \equiv \varphi[z = 0] \equiv \varphi$, pričom $\varphi[z = 0]$ je práve $\varphi \setminus z$, takže $\varphi \setminus z$ je identické s φ.

Ukázali sme teda, že (φ, z) je pozitívnou inštanciou „ZPM1“ práve keď (φ, C, z) je negatívnou inštanciou problému „Odstránenie premennej“. Problém „Odstránenie premennej“ je teda coNP úplný. □

Poznámka 6.2.5. Po takomto vyškrtnutí premennej sa z nikde v danej formule vyskytovať nebude, ale formulu budeme stále považovať za reprezentáciu funkcie na nezmenenej sade premenných.

Odstránenie klauzule

Inštancia: Matched CNF φ, klauzula $C \in \varphi$.

Otázka: Je $\psi = \varphi \setminus \{C\}$ ekvivalentné φ?

Veta 6.2.6. Problém „Odstránenie klauzule“ je CoNP úplný.

Dôkaz. Náležitosť do coNP je zrejmá. Certifikátom pre zápornú odpoveď je vektor, na ktorom sa φ a $\varphi \setminus C$ líšia.

Pri dôkaze coNP ťažkosti budeme postupovať prakticky rovnako ako pri predchádzajúcom dôkaze. Prevedieme negáciu problému „ZPM1“ na problém „Odstránenie klauzule“.

Nech (φ, z) je inštancia problému „ZPM1“ (z sa vo φ vyskytuje práve raz). O tomto probléme vieme, že je NP úplný. Ukážeme, že klauzulu obsahujúcu z môžeme z φ odstrániť práve vtedy, keď φ na z nezávisí.

Inštanciu (φ, z) prevedieme na inštanciu (φ, C), kde C je práve tá klauzula φ, ktorá obsahuje z.

Nech φ na z závisí. Keď odstránime klauzulu C obsahujúcu z, výsledok na z závisieť nebude, takže nemôže reprezentovať rovnakú funkciu ako φ.

Ak φ na z nezávisí, tak $\varphi[z = 1] \equiv \varphi[z = 0] \equiv \varphi$, pričom $\varphi[z = 1]$ je práve $\varphi \setminus C$ (pre C obsahujúce z), takže $\varphi \setminus C$ je identické s φ.

□

Poznámka 6.2.7. Pre matched DNF sú dôkazy analogické.

Je implikát

Inštancia: Matched CNF φ a klauzula C (nie nutne z φ).

Otázka: Je C implikátom φ?

Veta 6.2.8. Problém „Je implikát“ je coNP úplný.

Dôkaz. Problém zrejme do coNP patrí. Certifikátom negatívnej odpovede je vektor, na ktorom bude φ jedničková, ale C nulové.

CoNP ťažkosť ukážeme prevodom z problému „Odstránenie klauzule“. Inštanciu (φ, C) problému „Odstránenie klauzule“ prevedieme na inštanciu $(\varphi \setminus C, C)$ problému „Je implikát“. Tvrdíme, že C môžeme z φ odstrániť práve vtedy, keď je implikátom $\varphi \setminus C$.

Ak je $\varphi \setminus C$ identické s φ, tak pre každé ohodnotenie, kde $\varphi \setminus C$ má hodnotu jedna, má hodnotu jedna aj φ, ktorá C obsahuje, a teda C je implikát $\varphi \setminus C$.

A $\varphi \setminus C$ nie je identické s φ, tak existuje nejaké ohodnotenie premenných, na ktorom sa $\varphi \setminus C$ líši od φ. Keďže $\varphi \setminus C$ je podmnožinou φ, tak na tomto ohodnotení premenných musí byť $\varphi \setminus C$ jednička a φ nula. Teda C je nulové a nie je implikátom $\varphi \setminus C$. □

Minimalizácia matched CNF na počet literálov

Inštancia: Matched CNF φ a číslo k

Otázka: Existuje CNF ψ ekvivalentná φ, ktorá má počet výskytov literálov najviac k?

Veta 6.2.9. Problém „Minimalizácia matched CNF na počet literálov“ je Σ_2^p úplný.

Dôkaz. Spomeňme si na vetu 4.1.1. Tá tvrdí, že minimalizácia CNF a minimalizácia DNF sú ekvivalentné. Dôkaz sa dá použiť aj pre matched CNF/DNF, takže stačí ukázať, že minimalizácia matched DNF na počet literálov je Σ_2 úplná.

Na to sa použije nezmenený Umansov dôkaz vety 5.2.1, ktorý dokazuje Σ_2 úplnosť problému minimalizácie na počet literálov pre ľubovoľné DNF. Inštancia Σ_2 úplného problému „Shortest Implicant Core 1“ sa tam prevádza na inštanciu problému „Minimalizácia počtu literálov v DNF“. Táto inštancie má tvar (φ'', k'), kde $\varphi'' = t_n w_1 w_2 w_3 \dots w_{m'} \vee \bigvee_{i=1}^{m} s'_i$, kde $s'_i = s_i w_1 w_2 \dots w_{i-1} w_{i+1} \dots w_{m'}$. (φ'', k') je ale aj inštanciou minimalizácie matched DNF (na počet literálov) a preto je minimalizácia matched DNF na počet literálov a tým aj minimalizácia matched CNF na počet literálov Σ_2 úplná.

Ukážeme, že φ'' z dôkazu vety 5.2.1 je matched. Spomenieme si, že $m' = \max(m, k+1)$, čiže $m' \geq m$. Každému termu s'_i môžeme teda priradiť premennú w_{i+1}, pričom v prípade, že $m = m'$, termu s_m priradíme premennú w_1, ktorá priradená nebola. Prvému termu priradíme ľubovoľnú premennú z t_n a máme párovanie, ktoré každému termu priraďuje jednu premennú, čím sme ukázali, že φ'' je matched. □

Minimalizácia matched CNF na počet klauzúl

Inštancia: Matched CNF φ a číslo k

Otázka: Existuje CNF ψ ekvivalentná φ, ktorá má najviac k klauzúl?

Veta 6.2.10. Problém „Minimalizácia matched CNF na počet klauzúl“ je Σ_2^p úplný.

Dôkaz. Opäť sa odvoláme na vetu 4.1.1, ktorá tvrdí, že minimalizácia DNF a CNF je ekvivalentná a ukážeme, že minimalizácia matched DNF na počet termov je Σ_2 úplná.

Použijeme našu verziu Umansovho dôkazu vety 5.3.1. Prevádzame tam inštanciu Σ_2 úplného problému „Shortest Implicant Core 2“ na inštanciu problému „Minimalizácia počtu termov v DNF“. V pôvodných Umansových dôkazoch formula v tejto inštancii nebola matched. V našej verzii dôkazu ale matched je.

Ukážeme, že inštancia problému „Minimalizácia počtu termov v DNF“, na ktorú sme previedli inštanciu Σ_2 úplného problému „Shortest Implicant Core 2“ v dôkaze vety 5.3.1 obsahuje matched DNF. Formula v inštancii problému „Minimalizácia počtu termov v DNF“ je $\varphi' = \bigvee_{i=1}^{m} s'_i \vee \bigvee_{i=1}^{m} \bigvee_{j=1}^{n} u_{i,j}$, kde $s'_i = s_i z_1 z_2 \dots z_{i-1} z_{i+1} \dots z_m$, $u_{i,j} = (p_i)\overline{x_j} z_1 z_2 \dots z_m$ a p_i je „i-ty term jednoznačnej DNF reprezentácie parity na nových premenných $a_1, a_2, \dots a_r$, kde ako r zvolíme $m \times n$“. V prvej časti φ' priradíme termu s'_i premennú $z_{(i+1) \mod m}$, čím v tejto časti vytvoríme párovanie. V druhej časti každý z $m \times n$ termov $u_{i,j}$ obsahuje všetkých $m \times n$ premenných a, preto každému z nich môžeme jednu z týchto premenných priradiť. [1] Tým sme vytvorili párovanie pre formulu φ'. Táto formula je matched DNF a spolu s príslušným $k' = m(k+1)$ ukazuje, že minimalizácia matched DNF na počet termov je Σ_2 úplná. Podľa vety 4.1.1 je preto aj problém „Minimalizácia matched CNF na počet klauzúl“ Σ_2 úplný. □

6.3 Otvorené problémy

Zaujímavou a otvorenou otázkou je, či pre matched formulu formula s ňou ekvivalentná, ale minimálna, bude opäť matched, a to pre obe miery. Na prvý pohľad to vyzerá nepravdepodobne, ale protipríklad sa nepodarilo nájsť. Pri minimalizácii na počet termov (klauzúl) je možné, že platí silnejšie tvrdenie: Pre funkciu f ak φ je DNF (CNF) formula, ktorá má najmenší možný počet termov (klauzúl) spomedzi formulí reprezentujúcich f a má menej termov (klauzúl) ako premenných, tak φ je matched. Bez minimality tvrdenie neplatí – formula $abc + d + ed + e$ nie je matched, ale term ed je možné vynechať bez zmeny funkcie a formula sa stane matched. Či sa dá minimalita nahradiť napríklad primalitou všetkých jej termov ostáva tiež otvoreným problémom.

[1] Toto je dôvod, prečo sme použili paritu na takom množstve premenných.

Literatúra

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 2009.

[2] John Franco and Allen Van Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Appl. Math.*, 125(2-3):177–214, 2003.

[3] William Gasarch. The P=?NP poll. *SIGACT News*, 33(2), 2002. In SIGACT news complexity theory column 36.

[4] Judy Goldsmith, Matthias Hagen, and Martin Mundhenk. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Inf. Comput.*, 206(6):760–775, 2008.

[5] Peter L. Hammer and Alexander Kogan. Optimal compression of propositional Horn knowledge bases: complexity and approximation. *Artif. Intell.*, 64(1):131–145, 1993.

[6] Rostislav Horčík. Matematická logika. `http://www2.cs.cas.cz/~horcik/Teaching/ML04.pdf`, 2007. Slidy k predmetu Matematická Logika na ČVUT.

[7] Dániel Marx. Complexity of unique list colorability. *Theor. Comput. Sci.*, 401(1-3):62–76, 2008.

[8] Christos M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[9] Willard V. Quine. A way to simplify truth functions. *The American Mathematical Monthly*, 62(9):627–631, 1955.

[10] Stefan Szeider. Generalizations of matched CNF formulas. *Annals of Mathematics and Artificial Intelligence*, 43(1-4):223–238, 2005.

[11] Christopher M. Umans. The minimum equivalent DNF problem and shortest implicants. In *Foundations of Computer Science, 1998. Proceedings.39th Annual Symposium on*, pages 556 –563, 8-11 1998.

[12] Christopher M. Umans. Hardness of approximating Σ_2^p minimization problems. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 465–474, Washington, DC, USA, 1999. IEEE Computer Society.

[13] Christopher M. Umans. *Approximability and completeness in the polynomial hierarchy*. PhD thesis, University of California, Berkeley, 2000. Chair-Papadimitriou, Christos H.

[14] Václav Vlček. Třídy booleovských formulí s efektivně řešitelným satem. Master's thesis, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, 2009.