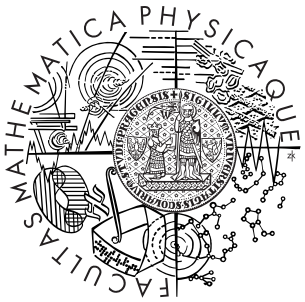
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Václav Vlček

Třídy Booleovských formulí s efektivně řešitelným SATem

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: doc. RNDr. Ondřej Čepek, Ph.D.
Studijní program: Informatika, Teoretická informatika

2009

Chtěl bych poděkovat vedoucímu práce za trpělivost a čas strávený při konzultacích a čtení průběžných verzí tohoto textu.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 30. července 2009 Václav Vlček

Obsah

Název práce: Třídy Booleovských formulí s efektivně řešitelným SATem
Autor: Václav Vlček
Katedra (ústav): Katedra teoretické informatiky a matematické logiky
Vedoucí diplomové práce: doc. RNDr. Ondřej Čepek, Ph.D.
e-mail vedoucího: Ondrej.Cepek@mff.cuni.cz

Abstrakt: Práce se zabývá třídami Booleovských formulí, pro které je problém splnitelnosti (SAT) řešitelný v polynomiálním čase. Zkoumá chování těchto tříd vzhledem k základním operacím s formulemi (komplementaci literálu, komplementaci proměnné, odebrání literálu nebo klauzule, částečné dosazení a spojení formulí). Dále se zabývá problémem rozpoznávání náležení formule do dané třídy, rozpoznávání splnitelnosti dané formule a vzájemnými vztahy těchto tříd vzhledem k inkluzi.

Klíčová slova: splnitelnost, třídy Booleovkých formulí, SAT

Title: Classes of Boolean Formulae with effectively solvable SAT
Author: Václav Vlček
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 work studies classes of Boolean formulae with polynomially solvable satsifiability problem (SAT). It investigates the behavior of these classes with respect to basic operation with formulae (variable and literal complementation, deletition of a literal or a clause, partial assignment and joining formulae). Furthermore the work studies recognition problems for a formula and a given class of functions, satisfiability testing, and mutual relations of the studied classes with respect to inclusion.

Keywords: satisfiability, classes of Boolean formulea, SAT

Kapitola 1

Úvod

Problém splnitelnosti (SAT) pro obecnou Booleovskou formuli v CNF je NP-úplný. Nicméně se vyskytuje v mnoha praktických úlohách jako je automatické dokazování, návrh plošných spojů, nebo řešení plánovacích problémů. Proto je snaha o jeho efektivní řešení. V této oblasti se vyskytují dva proudy – pravděpodobnostní řešení a řešení problému pro formule speciálního tvaru.

Tento text se bude zabývat druhým přístupem. Izolování určitých tříd totiž může pomoci při návrhu obecného řešiče problému SAT a to dvěma způsoby: v případě že rozpoznáme formuli z efektivně řešitelné třídy, můžeme jí vyřešit polynomiálním algoritmem (s garantovanou správností výsledku); některé myšlenky a výsledky lze také použít pro návrh heuristik pravděpodobnostního algoritmu.

Probereme zde jednotlivé třídy Booleovských formulí, pro které je znám polynomiální algoritmus na hledání splnitelnosti. Uvedeme základní výsledky týkající se vlastností formulí z těchto tříd, rozpoznávání a testování splnitelnosti. Prozkoumáme také vzájemný vztah tříd vzhledem k inkluzi. Ten má význam tehdy, chceme-li v obecném řešiči použít speciální algoritmus pro některé třídy. Víme-li, že jedna je podtřídou jiné, nemusíme testovat zvlášť náležení a splnitelnost pro obě třídy (stačí testovat pro tu větší).

V následující kapitole zavedeme potřebnou terminologii, exaktně popíšeme, čím se práce zabývá, a pojednáme o dvou přístupech používaných při řešení SATu – rezoluci a lineárním programování. V kapitole 3 pak probereme dobře známé třídy formulí (kvadratické, balanced, matched, Hornovské a rozšířené Hornovské). Kapitola 4 pak zkoumá třídy SLUR, q-Horn a LinAut; většina vlastních výsledků se nachází právě v této kapitole. V závěrečné části je přehledné shrnutí vzájemných vztahů zkoumaných tříd vzhledem k inkluzi a odvozených vlastností o uzavřenosti tříd vzhledem k operacím s formulemi.

Kapitola 2

Terminologie a metody

Nejdříve definujeme základní pojmy, které nám umožní říci, o čem bude tento text pojednávat. Následovat bude stručný popis výsledků ze dvou oblastí – rezoluční metody a řešení úloh lineárního programování – hojně se vyskytujících v řešeních zkoumaného problému.

2.1 Základní pojmy a popis problému

V našem textu budeme rozlišovat mezi *Booleovskou funkcí*, což je zobrazení ve smyslu teorie množin, a *Booleovskou formulí*, slovem ve formálním jazyce výrokové logiky. Tyto pojmy jsou definovány nasledovně:

Definice 2.1.1. *Booleovská funkce n proměnných* je zobrazení $f : \{0,1\}^n \to \{0,\ 1\}$. Hodnotu 1 chápeme jako pravdu (také označovanou jako true nebo T), hodnotu 0 jako nepravdu (false, F).

Prvku $x \in \{0,1\}^n$ budeme říkat *ohodnocení Booleovské funkce*. Pokud bude navíc $f(x) = 1$, budeme mluvit o *splňujícím ohodnocení*. Ohodnocení, pro než platí $f(x) = 0$ budeme označovat jako *falsifikující*.

Definice 2.1.2. Nechť V je neprázdná množina proměnných. *Výroková* (též *Booleovská) formule* je slovo nad abecedou $\Sigma = V \cup \{\neg, \&, \vee, \to, \leftrightarrow, (,)\}$ splňující následující pravidla:

1. Každá proměnná $x \in V$ je výrokovou formulí.
2. Jsou-li A a B výrokové formule, jsou i $\neg A$, $(A\&B)$, $(A \vee B)$, $(A \to B)$ a $(A \leftrightarrow B)$ výrokové formule.
3. Každá formule vznikne konečným počtem aplikací pravidel 1. a 2.

Tímto jsme definovali jazyk výrokové logiky (syntax), nyní slovům tohoto jazyka přiřadíme význam (definujeme sémantiku).

Definice 2.1.3. Zobrazení $v : V \to \{0,1\}$ nazýváme *pravdivostním ohodnocením proměnných*. Toto zobrazení v jednoznačně rozšíříme do *pravdivostního ohodnocení formule* $\bar{v}$ indukcí podle složitosti formule:

- $\bar{v}(x) = v(x)$, je-li $x \in V$

- $\bar{v}(\neg A) = \begin{cases} 0, & \text{je-li } \bar{v}(A) = 1 \\ 1, & \text{je-li } \bar{v}(A) = 0 \end{cases}$ *(negace)*
- $\bar{v}(A\&B) = \begin{cases} 1, & \text{je-li } \bar{v}(A) = \bar{v}(B) = 1 \\ 0, & \text{jinak} \end{cases}$ *(konjunkce)*
- $\bar{v}(A \vee B) = \begin{cases} 0, & \text{je-li } \bar{v}(A) = \bar{v}(B) = 0 \\ 1, & \text{jinak} \end{cases}$ *(disjunkce)*
- $\bar{v}(A \to B) = \begin{cases} 0, & \text{je-li } \bar{v}(A) = 1 \text{ a } \bar{v}(B) = 0 \\ 1, & \text{jinak} \end{cases}$ *(implikace)*
- $\bar{v}(A \leftrightarrow B) = \begin{cases} 1, & \text{je-li } \bar{v}(A) = \bar{v}(B) \\ 0, & \text{jinak} \end{cases}$ *(ekvivalence)*

Obdobně jako dříve definujeme *splňující ohodnocení formule* F jakožto rozšíření zobrazení $v : V \to \{0,1\}$ na zobrazení $\bar{v}$ způsobem uvedeným výše takové, že platí $\bar{v}(F) = 1$ (analogicky *falsifikující ohodnocení*).

Příklad 2.1.4. Na příkladu ukážeme, že každou Bool. funkci $f : \{0,\ 1\}^n \to \{0,1\}$ lze vyjádřit výrokovou formulí F nad proměnnými $\{x_1, x_2, \ldots, x_n\}$.

Vezměme si funkci tří proměnných definovanou následující tabulkou:

x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Formuli F sestavíme po částech. Soustředíme se pouze na řádky ve kterých je funkční hodnota rovna jedné. Pro každý takový řádek vytvoříme formuli jako konjunkci všech proměnných tak, že za proměnnou v jejímž sloupci je 0 použijeme poměnnou v negované formě a za proměnou, kde je 1 použijeme proměnnou v přímé formě. Tak například pro řádek | 1 | 0 | 1 || 1 | vznikne formule $(x_1\&\neg x_2\&x_3)$. Požadovanou formuli odpovídající Booleovské funkci f zadané tabulkou pak získáme jako disjunkci všech takto vzniklých formulí. V našem případě $F = (\neg x_1\&\neg x_2\&\neg x_3) \vee (\neg x_1\&\neg x_2\&x_3) \vee (\neg x_1\&x_2\&\neg x_3) \vee (x_1\&\neg x_2\&x_3) \vee (x_1\&x_2\&\neg x_3)$.

Tímto způsobem vždy vznikne formule specifického tvaru (v tzv. disjunktivní normální formě – DNF). Tento tvar definujeme, abychom ujasnili terminologii. Současně definujeme konjunktivní normální formu (CNF). Z tvrzení, které bude následovat, pak vyplyne, že každá Booleovská funkce lze vyjadřit i v CNF.

Definice 2.1.5 (normální tvary)**.**

- *Literálem* nazveme proměnnou nebo její negaci.
- *Termem* označíme konečnou konjunkci literálů, která neobsahuje současně literál i jeho negaci.

- Formule je v *disjunktivní normální formě* (zkráceně *DNF*), jestliže je konečnou disjunkcí termů.
- *Klauzulí* nazveme konečnou disjunkci literálů neobsahující současně literál i jeho negaci.
- Formule je v *konjunktivní normální formě (CNF)*, jestliže je konečnou konjunkcí klauzulí.

Tvrzení 2.1.6 (de Morganova pravidla)**.** *Nechť L je konečná množina literálů. Formule $F_1 = \neg \bigwedge_{l \in L} l$ a formule $F_2 = \bigvee_{l \in L} \neg l$ vyjadřují tutéž Booleovskou funkci (jsou* ekvivalentní *— to značíme symbolem $\equiv$).*

Důkaz. Nechť v je splňující ohodnocení formule F_1 (tj. $\bar{v}(F_1) = 1$). Platí $\bar{v}(F_1) = 1 \Leftrightarrow \bar{v}(\neg \bigwedge_{l \in L} l) = 1 \Leftrightarrow \bar{v}(\bigwedge_{l \in L} l) = 0 \Leftrightarrow$ alespoň pro jeden literál $l \in L$ je $\bar{v}(l) = 0 \Leftrightarrow$ alespoň pro jeden literál $l \in L$ je $\bar{v}(\neg l) = 1 \Leftrightarrow \bar{v}(\bigvee_{l \in L} \neg l) = 1 \Leftrightarrow \bar{v}(F_2) = 1$. □

Příklad 2.1.7. Nyní se vrátíme k příkladu 2.1.4 a ukážeme na téže formuli vyjádření v CNF. Všimněme si, že dvojitá negace se vyruší (tj. $\neg\neg F \equiv F$). Vytvoříme tedy ze zadané Booleovské funkce f funkci $\bar{f}$ tak, že zaměníme funkční hodnoty 1 za 0 a naopak. K ní vytvoříme odpovídající formuli v DNF $\bar{F} = (\neg x_1 \& x_2 \& x_3) \vee (x_1 \& \neg x_2 \& \neg x_3) \vee (x_1 \& x_2 \& x_3)$ ekvivalentní s $\neg F$.

Nyní použijeme de Morganova pravidla: $\neg \bar{F} \equiv (x_1 \vee \neg x_2 \vee \neg x_3) \& (\neg x_1 \vee x_2 \vee x_3) \& (\neg x_1 \vee \neg x_2 \vee \neg x_3)$, což je formule v CNF vyjadřující f.

Poznámka 2.1.8. Ve zbytku tohoto textu se omezíme pouze na konjunktivní normální tvar (CNF). Protože při této úmluvě nebude hrozit záměna CNF a DNF, můžeme používat také množinový popis formulí. Formuli pak budeme chápat jako množinu klauzulí a klauzuli jako množinu jejích literálů.

V této práci se budeme zabývat algoritmickým testováním splnitelnosti formule. Speciálně nás budou zajímat ty třídy formulí, pro které lze tento problém řešit rychle pomocí deterministického algoritmu. Proto je nyní třeba zavést výpočetní model a přesněji specifikovat, co znamená řešit „rychle“.

Definice 2.1.9 (výpočetní model)**.** *Turingův stroj M s k páskami* je pětice $M = (Q, \Sigma, \delta, q_0, F)$, kde

- Q je neprázdná konečná množina stavů,
- Σ je neprázdná konečná pásková abeceda obsahující symbol pro prázdné políčko,
- $\delta : Q \times \Sigma^k \to Q \times \Sigma^{k-1} \times \{R, N, L\}^k$ je (přip. částečná) přechodová funkce,
- $q_0 \in Q$ je počáteční stav,
- $F \subset Q$ je množina přijímajících stavů.

Všechny pásky jsou jednostranně potenciálně nekonečné, skládají se z buňek, ve kterých může být zapsan libovolný symbol páskové abecedy, nebo mohou být prázdné.

Konfigurací Turingova stroje rozumíme aktuální stav výpočtu popsaný pomocí: stavu ve kteréme se Turinguv stroj nachazí, popsáných částí pásek a pozic hlav na páskách.

V *počáteční konfiguraci* je Turingův stroj v počátečním stavu q_0. První *(vstupní)* páska na svém začátku obsahuje vstupní slovo, ostatní *(pracovní)* pásky jsou prázdné. Hlavy se na všech páskách nachází na první buňce.

Jeden *krok výpočtu* je popsán přechodovou funkcí δ: ke stavu q a obsahu jednotlivých pásek v políčkách pod hlavami získáme jejím užitím nový stav, pokyn pro hlavy nad pracovními páskami, co mají zapsat na svou aktuální pozici a kam se následně mají posunout (pokyny jsou pro všechny hlavy, pro každou zvlášť).

Výpočtem Turingova stroje pak rozumíme posloupnost konfigurací začínající v počáteční konfiguraci, kde mezi dvěma po sobě následujícími konfiguracemi lze přejít pomocí přechodové funkce. Tato posloupnost může být buďto nekonečná, nebo pro poslední konfiguraci v posloupnosti není definována přechodová funkce, případně obsahuje právě jeden koncový stav (poslední prvek posloupnosti). Říkáme, že Turingův stroj *přijímá slovo* w, jestliže výpočet končí a to v přijímajícím stavu.

Množinu slov přijímaných strojem M *(jazyk)* označíme $L(M)$.

Definice 2.1.10 (časová složitost)**.** Nechť Turingův stroj M provede nad každým vstupem délky n nejvýše $T(n)$ kroků, než se zastaví, pak říkáme, že M má *časovou složitost* $T(n)$[1].

Definice 2.1.11 (třída P)**.** Jazyk L patří do třídy P, jestliže existuje polynom $p : \mathbb{N} \to \mathbb{N}$ a Turingův stroj M takový, že $L = L(M)$ a M má časovou složitost $T(n) \leq p(n)$ pro všechna n.

Poznámka 2.1.12. Výpočetní model byl vybrán tak, jak je obvyklé. Pokud připustíme plastnost známé „silné Church–Turingovy teze“[2], tak na konkrétní volbě modelu vlastně nezáleželo (získali bychom stejnou třídu P).

Jak je běžné, budeme algoritmy popisovat pseudokódy a předpokládat, že jsme schopni příslušný Turingův stroj zkonstruovat. Dále budeme uvádět časovou složitost tak, aby co nejvíce odpovídala reálnému výpočetnímu stroji (budeme pro tyto potřeby uvažovat stroj RAM). K odhadům časové složitosti algoritmů budeme používat „O“-notaci. Pro úplnost uvádíme definici symbolu O, ostatní používané symboly (o, ω, Ω, Θ) čtenář najde v libovolné knize zabývající se výpočetní složitostí (např. [1]).

Definice 2.1.13. Funkce $f : \mathbb{N} \to \mathbb{N}$ je asymptoticky menší než $g : \mathbb{N} \to \mathbb{N}$, píšeme $f = O(g)$, jestliže $(\exists c \in \mathbb{N})(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(f(n) \leq c \cdot g(n))$.

Nyní ještě zmíníme několik faktů o nedeterministických výpočtech, polynomiálních transformacích a třídách NP a co-NP.

Poznámka 2.1.14. Turingův stroj dle uvedené definice má vždy následující konfiguraci určenu jednoznačně – plyne z požadavku, že δ je funkce. Když od tohoto požadavku upustíme (formálně bude nyní δ funkce do potenční množiny $\mathscr{P}(Q \times \Sigma^{k-1} \times \{R, N, L\}^k)$), získáme *nedeterministický Turingův stroj*.

Pak musíme také náležitě upravit definici přijímání: nedeterministický Turingův stroj přijímá slovo w, jestliže existuje přijímající výpočet (tj. posloupnost konfigurací začínající počáteční konfigurací a končící v přijímajícím stavu, taková, že mezi dvěma po sobě jdoucími konfiguracemi lze přejít pomocí přechodové funkce).

[1] Přesněji řečeno se jedná o časovou složitost v nejhorším případě, ale jiná nás zajímat nebude.

[2] Silná Church–Turingova teze říká, že každý „rozumný“ výpočetní model lze s polynomiálním zpomalením simulovat na Turingově stroji.

A také upravit definici času výpočtu jako délku nejkratšího z přijímajících výpočtů, pokud existuje.

Definice 2.1.15. Nechť L je jazyk nad abecedou Σ a L' je jazyk nad abecedou Σ'. Pak funkci $f : \Sigma^* \to \Sigma'^*$ nazveme *polynomiální transformací*, pokud existuje deterministický Turingův stroj takový, že pro každé slovo $w \in \Sigma^*$ na vstupní pásce spočítá v polynomiálním čase slovo $f(w) \in \Sigma'^*$ a zapíše ho na první pracovní pásku a pro $f(w)$ platí $w \in L \Leftrightarrow f(w) \in L'$.

Definice 2.1.16 (třídy NP a co-NP)**.** *NP* je třída těch jazyků L, pro které existuje nedeterministický Turingův stroj pracující v polynomiálním čase přijímající právě slova z L.

Třída *co-NP* je třída jazyků $L \subseteq \Sigma^*$ takových, že doplněk jazyka L, tj. $\bar{L} = \{w \in \Sigma^* : w \notin L\}$, je přijímán nedeterministickým Turingovým strojem v polynomiálním čase.

Poznámka 2.1.17 (certifikátová definice)*.* Nedeterministický výpočet si můžeme znázornit stromem, jehož vrcholy odpovídají konfiguracím a hrany odpovídají přechodům mezi konfiguracemi, v kořeni je počáteční konfigurace. Pro polynomiální výpočet (cestu která začíná v kořeni, je polynomiálně dlouhá vzhledem k délce vstupu a končí přijetím) pak můžeme vytvořit polynomiálně dlouhé slovo kódující vybranou možnost přechodu v uzlech s více následníky.

Ve skutečnosti lze třídu NP definovat také následovně: Jazyk L patří do třídy NP, jestliže existuje polynom p a pro každé slovo $w \in L$ existuje „certifikát“ délky nejvýše $p(|w|)$, takové, že s jeho pomocí lze na deterministickém Turingově polynomiálně stroji ověřit, že $w \in L$.

Definice 2.1.18. Jazyk L je NP-těžký, pokud pro libovolný jazyk ze třídy NP existuje polynomiální transformace na L.

Jazyk L je NP-úplný, jestliže je ve třídě NP a je NP-těžký. Obdobně pro co-NP-těžkost a co-NP-úplnost.

Nyní můžeme přesně vyslovit, čemu se budeme v tomto textu věnovat. Za tím účelem definujeme problém splnitelnosti Booleovských formulí (SAT).

Definice 2.1.19. Problém SAT je definován následovně:
Instance: Formule F v konjunktivně normální formě.
Otázka (podmínka pro náležení do jazyka)**:** Existuje ohodnocení proměnných v formule F takové, že $\bar{v}(F) = 1$?

Je známo, že problém SAT je ve své obecnosti NP-úplný. Není však zatím známo, zda jej lze řešit polynomiálně i deterministicky (není rozřešená otázka vztahu tříd P a NP). Protože se vyskytuje i v praktických problémech, je snaha jej řešit polynomiálně alespoň pro nějakou podtřídu formulí. Naším cílem bude shrnout známé výsledky o třídách formulí, pro které je znám polynomiální algoritmus na rozhodování splnitelnosti a zjistit, v jakém jsou vzájemném vztahu vzhledem k inkluzi.

Poznámka 2.1.20*.* Vedle snahy nacházet třídy s polynomiálním algoritmem testování splnitelnosti se rozvíjí také oblast pravděpodobnostních algoritmů na řešení tohoto problému, též se hledají možnosti, jak polynomiálně převést řešení SATu na jiný problém (například splňování omezujících podmínek – CSP) pro který jsou již vyvinuté řešící techniky (také ne obecně polynomiální). Těmto oblastem se věnovat nebudeme.

2.2 Rezoluční metoda

Dále se budeme zabývat rezoluční metodou, protože její myšlenky se využívají i pro řešení splnitelnosti formulí z některých tříd, jež budou následovat. Definujeme několik dalších pojmů. Tvrzení obsažená v této kapitole patří k základům problematiky booleovských funkcí a v obměně pro DNF je lze nalézt například v připravované knize [7].

Definice 2.2.1. Klauzule C je *implikátem formule* F, jestliže pro každé ohodnocení v platí $\bar{v}(C) = 0 \Rightarrow \bar{v}(F) = 0$.

Klauzule C je *primárním implikátem* F, pokud každá C' vzniklá z C vypuštěním libovolného jejího literálu již není implikát F.

Poznámka 2.2.2. Každá klauzule formule F je zároveň jejím implikátem. Navíc podle následujícího tvrzení můžeme implikáty přidávat k formuli, aniž bychom změnili reprezentovanou funkci. Cílem rezoluční metody je najít formuli, kde budou všechny implikáty primární.

Tvrzení 2.2.3. *Jestliže klauzule C je implikátem formule F, pak F i $C\&F$ reprezentuje tutéž Booleovskou funkci.*

Důkaz. Nechť v je ohodnocení proměnných formule F. Rozebereme dva případy podle hodnoty klazule C:

a) $\bar{v}(C) = 1$: Pak dle pravidla pro vyhodnocení konjunkce máme $\bar{v}(\mathrm{C}\ \&\ \mathrm{F}) = \bar{v}(C)\&\bar{v}(F)$. Víme že první člen je ohodnocen na 1. Podle definice konjunkce tak je $\bar{v}(\mathrm{C}\ \&\ \mathrm{F}) = \bar{v}(F)$.

b) $\bar{v}(C) = 0$: Z toho, že C je implikát, plyne, že také $\bar{v}(F) = 0$ (a platí tedy $\bar{v}(\mathrm{C}\ \&\ \mathrm{F}) = \bar{v}(F)$).

V obou případech platí $\bar{v}(\mathrm{C}\ \&\ \mathrm{F}) = \bar{v}(F)$). □

Definice 2.2.4. Nechť C je implikát F (tj. $\bar{v}(F) \leq \bar{v}(C)$ pro vš. v) a C' je primární implikát F splňující $\bar{v}(F) \leq \bar{v}(C') \leq \bar{v}(C)$ (uvědomme si, že to znamená, že všechny literály z C' jsou v C), pak říkáme, že C' *absorbuje* C.

Poznámka 2.2.5. Jestliže F reprezentuje funkci f a obsahuje všechny primární implikáty f, lze ostatní klauzule vypustit beze změny reprezentované funkce (jsou absorbovány nějakým z implikátů). Vzniklou formuli pak označíme jako *kanonickou CNF*.

Kanonickou reprezentaci lze najít pomocí rezoluční metody. Ta je založena na následujícím tvrzení.

Definice 2.2.6. Klauzule C_1 a C_2 mají *konflikt v proměnné* x, právě když jedna z nich obsahuje literál x a druhá literál $\neg x$.

Klauzule $C_1 = (A \vee x)$ a $C_2 = (B \vee \neg x)$ mají společnou *rezolventu* $A \vee B$, jestliže mají konflikt právě v jedné proměnné (v proměnné x).

Tvrzení 2.2.7. *Nechť $C_1 = (A \vee x)$ a $C_2 = (B \vee \neg x)$ jsou klauzule formule F a mají rezolventu $A \vee B$. Pak $A \vee B$ je implikátem formule F.*

Algoritmus 1 Rezoluční metoda

Vstup: Formule F v CNF

Výstup: Formule v kanonické CNF

1: $F := F$ po vynechání absorbovaných klauzulí
2: **while** (v F jsou klauzule C_1 a C_2 mající společnou rezolventu C_3 a ta není absorbována žádnou klauzulí z F) **do**
3: $\quad F := F \cup C_3$
4: $\quad F := F$ po vynechání absorbovaných klauzulí
5: **end while**
6: **return** F

Důkaz. Jestliže $\bar{v}(A \vee B) = 0$, pak je $\bar{v}(A) = \bar{v}(B) = 0$. Potom platí alespoň jedna z rovností $\bar{v}(A \vee x) = 0$ nebo $\bar{v}(B \vee \neg x) = 0$. A tedy $\bar{v}(F) = 0$, protože $A \vee x$ a $B \vee \neg x$ jsou implikáty. Celkem tedy máme požadovanou implikaci $\bar{v}(A \vee B) = 0 \Rightarrow \bar{v}(F) = 0$, která znamená, že $A \vee B$ je implikát. □

Tvrzení 2.2.8. *Algoritmus rezoluční metody se vždy zastaví a vrátí kanonickou CNF.*

Důkaz. Algoritmus je konečný, protože žádný implikát nebude přidán dvakrát – každý přidaný implikát buď zůstane ve formuli až do konce výpočtu, nebo je později absorbován jiným přidaným implikátem a už nemůže být přidán znovu.

Sporem ukážeme, že každý primární implikát je přidán do výsledné formule. Nechť algoritmus vrátí formuli $F = C_1 \& C_2 \& \ldots \& C_k$ a C je primární implikát neobsažený ve formuli F (tj. $(\forall i)(C \neq C_i)$). Definujme množinu $I_C = \{C' : (C \subseteq C') \& (\forall i)(C_i \not\subseteq C')\}$ všech implikátů absorbovaných primárním implikátem C, jež nejsou absorbovány žádným implikátem formule F. I_C je neprázdná, protože obsahuje C. Nechť C^* je nejdelší klauzule z množiny I_C, a n je počet všech proměnných formule F. Rozebereme dva případy:

a) $|C^*| = n$ (obsahuje všechny proměnné): $\bar{v}(C^*) = 0$ (tj. žádný literál není splněn) $\Rightarrow (\forall i)(\bar{v}(C_i) = 1)$, protože každá C_i má s C^* konflikt (a tedy splněný literál) díky podmínce $(\forall i)(C_i \not\subseteq C')$ v definici množiny I_C. To znamená, že $\bar{v}(F) = 1$.

   Také ale (podle podmínky $C \subseteq C'$) musí platit $\bar{v}(C^*) = 0 \Rightarrow \bar{v}(C) = 0$, což znamená, že $\bar{v}(F) = 0$, protože C je implikát. Tato hodnota dává spor s předchozí úvahou.

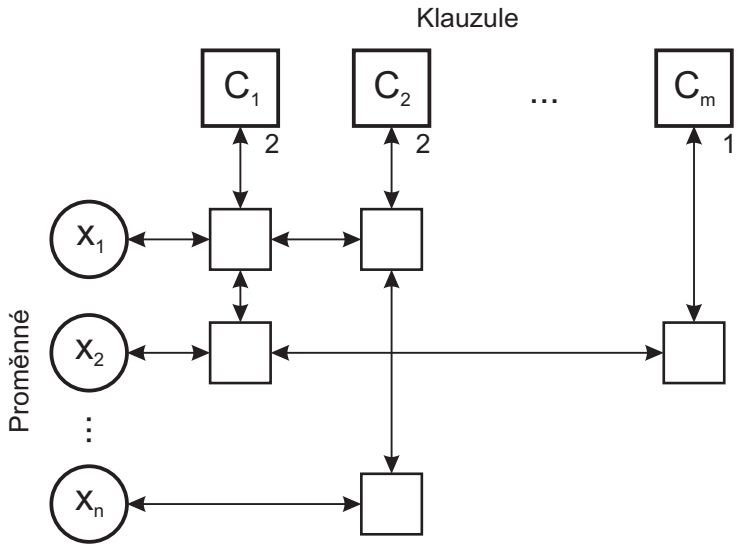
b) $|C^*| < n$: Nechť literály $x, \neg x$ nejsou v C^*, tzn. $C^* \vee x$ ani $C^* \vee \neg x$ nejsou v I_C. To znamená, že pro ně neplatí podmínka $(\forall i)(C_i \not\subseteq C')$ z definice I_C. Existují tedy i a j takové, že $C_i \subseteq C^* \cup \{x\}$ a $C_j \subseteq C^* \cup \{\neg x\}$. Protože $C^* \in I_C$, je $x \in C_i$ a $\neg x \in C_j$. Společná rezolventa C_i a C_j je pak v množině I_C a není absorbována žádným implikátem C_k, což je spor.

□

Rezoluční metoda v obecném případě nepracuje polynomiálně vzhledem k délce vstupní formule. Z toho, že ji lze použít k testování splnitelnosti libovolné Booleovské formule, plyne, že vygenerování kanonické CNF je NP-těžký problém: formule F

je nesplnitelná, právě když pro každé ohodnocení v je $\bar{v}(F) = 0$, taková formule má ale jako implikant identickou nulu (prázdnou klauzuli). Protože identická nula absorbuje libovolnou klauzuli, je zároveň kanonickou reprezentací formule F. Celkem tedy máme, že formule F je splnitelná, právě když rezoluční metoda nevrátí jako výsledek formuli s jedinou nulovou klauzulí.

Často se jako podprogram používá omezená varianta – *jednotková rezoluce*. Ta provádí odvozování rezolventy pouze tehdy, když je jedna z klauzulí jednotková (je tvořená pouze jedním literálem) – ta vynucuje hodnotu pro danou proměnnou. V případě, kdy rezoluci používáme pro potřeby testování splnitelnosti, se využívá ještě upravená varianta – *propagace jednotkových klauzulí* – viz algoritmus unitprop (na straně 14). Tento algoritmus lze implementovat v lineárním čase vzhledem k délce formule. K tomu poslouží reprezentace formule pomocí struktury z dvousměrných seznamů uvedená na obrázku (v každém průběhu while-cyklu vypadne jeden celý řádek a alespoň jeden celý sloupec; protože struktura obsahuje právě tolik prvků, kolik je ve formuli literálů a každý je použit nejvýše jednou a to s konstantními nároky na čas, je celá procedura lineární).



Obrázek 2.1: Struktura pro reprezentaci formule v algoritmu unitprop

Pro studium vlastností rezoluce je někdy výhodné použít tzv. rezoluční graf (použijeme ho v části o třídě SLUR). Ten je definován následujícím způsobem.

Definice 2.2.9 (rezoluční graf)**.** *Rezoluční graf pro formuli F* je neorientovaný graf, kde možina vrcholů odpovídá množině klauzulí ve formuli F a mezi dvěma vrcholy vede hrana, právě když mají konflikt právě v jedné proměnné.

2.3 Lineární programování

Testování splnitelnosti je možno polynomiálně převést na úlohu celočíselného lineárního programování (ta je tak NP-těžká). Protože pro lineární programování s reálnými (nebo racionálními) čísly existují polynomiální algoritmy, objevily se myšlenky zavést třídu formulí tak, že formule patří do dané třídy, pokud lze její

Algoritmus 2 unitprop

Vstup: Formule F v CNF

Výstup: Formule po provedení propagace jednotkových klauzulí a vynucené částečné ohodnocení

```
1: t := „prázdné částečné ohodnocení“
2: while (v F je jednotková klauzule l) do
3:   if F obsahuje prázdnou klauzuli then
4:     return (F, t)
5:   end if
6:   t := t ∪ „ohodnocení proměnné literálu l tak, aby byl l splněn“
7:   for all C je klauzule F do
8:     if l se vyskytuje v C then
9:       F := F \ {C}
10:    else if ¬l se vyskytuje v C then
11:      F := (F \ {C}) ∪ ({C \ {l}})
12:    end if
13:  end for
14: end while
15: return (F, t)
```

splnitelnost rozhodnout zaokrouhlením reálného řešení (případně se zaokrouhlování používá při postupném získávání částečných ohodnocení).

Protože samotná problematika polynomiálního řešení úlohy lineárního programování přesahuje rámec tohoto textu, pouze řekneme, co je to úloha lineárního programování a jak se na ní problém splnitelnosti typicky převádí. Pro podrobnosti o polynomiálním řešení této úlohy i o lineárním programování obecně může čtenář nahlédnout do knihy [15].

Definice 2.3.1. Pro vektory $\vec{x}, \vec{y} \in \mathbb{Q}^n$ (resp. $\mathbb{R}^n$, $\mathbb{Z}^n$) definujeme částečné uspořádání $\vec{x} \leq \vec{y}$, právě když $(\forall i \in \{1, \ldots, n\})(x_i \leq y_i)$.

Definice 2.3.2. Nechť A je matice typu $m \times n$ nad $\mathbb{R}$, $\vec{b} \in \mathbb{R}^m$ a $\vec{c}, \vec{x} \in \mathbb{R}^n$. Potom úlohou lineárního programování je najít $\vec{x^*}$ tak, že $\vec{c}^T\vec{x^*} = \min\{\vec{c}^T\vec{x} : A\vec{x} = \vec{b}, \vec{x} \geq \vec{0}\}$.

Libovolné $\vec{x}$ splňující podmínky $A\vec{x} = b, \vec{x} \geq \vec{0}$ nazveme přípustným řešením, libovolné $\vec{x^*}$ nazveme optimálním řešením.

Úlohu lineárního programování lze formulovat i jako maximalizační, připustit ve výrazu $A\vec{x} = b$ nerovnost nebo použít více nerovností na danou proměnnou. Tyto varianty lze však převést na uvedený základní tvar vhodnou změnou znaménka a přidáním proměnných a nerovností pro ně.

Poznámka 2.3.3. V případě, že budeme požadovat, aby $\vec{x} \in \mathbb{Z}$, úloha se stane NP-těžkou, protože na ni bude možno převést problém splnitelnosti (SAT) polynomiální transformací:

Definice 2.3.4. *Incidenční maticí* M_F pro formuli F nazveme matici typu $m \times n$ (kde m je počet klauzulí a n počet proměnných) definovanou následovně:

$$(M_F)_{i,j} = \begin{cases} 1, & \text{jestliže } i\text{-tá klauzule } F \text{ obsahuje } j\text{-tou proměnnou v přímé formě} \\ -1, & \text{jestliže } i\text{-tá klauzule } F \text{ obsahuje } j\text{-tou proměnnou v negované} \\ 0, & \text{jinak} \end{cases}$$

Tvrzení 2.3.5. *Formule F je splnitelná, právě když má úloha najít x^* maximalizující $\vec{e}^T\vec{x}$ za podmínek $M_F \cdot \vec{x} \geq 2\vec{e} - \vec{L}, -1 \leq \vec{x} \leq 1, \vec{x} \in \mathbb{Z}^n$ alespoň jedno přípustné řešení (kde $\vec{e}$ je vektor samých jedniček a $\vec{L}$ je vektor délek jednotlivých klauzulí).*

Důkaz. Nejdříve několik vysvětlujících poznámek: Podmínky $-1 \leq \vec{x} \leq 1, \vec{x} \in \mathbb{Z}^n$ zajišťují, že případná řešení jsou z množiny $\{-1, 0, 1\}$. Výraz $\vec{e}^T\vec{x}$ vyjadřuje, že maximalizujeme počet proměnných nastavených na hodnotu 1 (odpovídá součtu všech složek vektoru $\vec{x}$).

Existuje vzájemně jednoznačná korespondence mezi splňujícími ohodnoceními proměnných formule F a přípustnými řešeními uvedené úlohy: Pro ohodnocení v vezmeme vektor $\vec{x}$, jehož i-tá složka je rovna 1 (resp. -1), právě když je i-tá proměnná ohodnocena true (resp. false). Hodnota 0 odpovídá tomu, že za proměnnou nedosazujeme.

Výsledek součinu $M_F \cdot \vec{x}$ odpovídá rozdílu počtu splněných literálů v klauzuli a počtu nesplněných literálů. Podmínka $M_F \cdot \vec{x} \geq 2\vec{e} - \vec{L}$ pak požadavku, aby byl splněn alespoň jeden literál v každé klauzuli. □

Poznámka 2.3.6. Někdy se také používá varianta s $\vec{x} \in \{0, 1\}^n$. Pak se nerovnost změní na $M_F \cdot \vec{x} \geq \vec{e} - \vec{L}$, protože se neodečítají nesplněné literály.

Kapitola 3

Známé třídy s polynomiálně řešitelným SATem

Nyní postupně probereme třídy, pro něž je znám polynomiální algoritus rozhodující problém splnitelnosti. Uvedeme přehled známých výsledků a na závěr přehledný diagram vzájemných vztahů vzhledem k inkluzi.

3.1 Kvadratické formule

Definice 3.1.1. Formule F je *kvadratická*, jestliže každá její klauzule obsahuje nejvýše dva literály. Pokud každá klauzule obsahuje právě dva literály, hovoříme o *ryze kvadratické* formuli.

Pozorování 3.1.2. *Třída kvadratických formulí je uzavřená na částečné dosazení, odebrání literálu i klauzule, komplementaci literálu i proměnné a spojení dvou formulí.*

Důkaz. Žádná z uvedených operací nezvyšuje počet literálů v klauzuli. □

Třídu kvadratických formulí lze jednoduše rozpoznávat v lineárním čase — stačí ověřit počet literálů v klauzulích.

K testování splnitelnosti uvedeme lineární algoritmus, který pochází z článku [3]. Doplníme také poznámku o algoritmu založeném na propagaci jednotkových klauzulí (konkrétní lineární algoritmus založený na tomto přístupu lze najít v článku [10]).

Dále budeme předpokládat, že formule F je ryze kvadratická. To není příliš omezující předpoklad, protože jednotkové klauzule vynucují hodnotu proměnné a ryze kvadratickou formuli lze z obecné kvadratické získat např. procedurou unitprop.

Poznámka 3.1.3. Každou kvadratickou klauzuli můžeme vyjádřit pomocí implikace a to dvěma způsoby: $(l_1 \vee l_2) \equiv (\neg l_1 \to l_2) \equiv (\neg l_2 \to l_1)$.

Definice 3.1.4 (implikační graf)**.** Pro kvadratickou formuli F s proměnnými V definujeme implikační (orientovaný) graf $G_F = (V_F, E_F)$ následujícím způsobem: množina vrcholů je tvořena všemi literály nad proměnnými V, tj. $V_F = \{x, \neg x : x \in V\}$; pro každou klauzuli $(l_1 \vee l_2)$ formule F přidáme do E_F hrany $(\neg l_1, l_2)$ a $(\neg l_2, l_1)$ — ty odpovídají oběma implikacím uvedeným výše.

V implikačním grafu každá hrana odpovídá vyjádření některé klauzule z formule pomocí implikace (pro jednu klauzuli jsou v grafu dvě hrany/implikace). Fomule je splněna, právě když je splněna každá klauzule (a klauzule je ekvivalentní s oběma implikacemi). Pro incidenční graf zavedeme funkci ohodnocující jeho vrcholy logickými hodnotami. Tato funkce bude odpovídat splňujícímu ohodnocení formule za následujících zřejmých podmínek.

Tvrzení 3.1.5. [3] *Ohodnocení $o : V_F \to \{0, 1\}$ odpovídá splňujícímu ohodnocení formule v, právě když:*

a) pro každé $x \in V$ je $v(x) = o(x) \neq o(\neg x) = \bar{v}(\neg x)$ (proměnná a její komplement mají opačné hodnoty).

b) jestliže z x do y vede hrana a $o(x) = 1$, pak také $o(y) = 1$ (požadavek na splnění implikace $x \to y$).

Definice 3.1.6 (silně souvislá komponenta)**.** Množina vrcholů $U \subseteq V_F$ tvoří silně souvislou komponentu, právě když pro každé $u, v \in U$ existuje orientovaná cesta z u do v a U je maximální množina s touto vlastností (tj. nelze k ní přidat další vrcholy).

Pozorování 3.1.7. [3] *Pokud v G_F existuje orientovaná cesta z vrcholu u do vrcholu v, pak díky dualitě definice existuje také cesta z $\neg v$ do $\neg u$. Z toho plyne, že pokud U tvoří silně souvislou komponentu, pak také $\bar{U} = \{\neg v : v \in U\}$ tvoří silně souvislou komponentu.* □

Tvrzení 3.1.8. [3] *Formule F je splnitelná, právě když v G_F žádný vrchol x není ve stejné silně souvislé komponentě jako vrchol odpovídající jeho komplementu $\neg x$.*

Důkaz. Jestliže x a $\neg x$ leží ve stejné silně souvislé komponentě (existují orientované cesty — řetězce z implikací — oběma směry), tak každé ohodnocení poruší alespoň jednu z podmínek v tvrzení 3.1.5.

Naopak pokud jsou $(\forall x \in V)$ je x a $\neg x$ v různých silně souvislých komponentách incidenčního grafu, sestrojíme splňující ohodnocení následujícím způsobem:

1. Setřiď komponenty souvislosti v topologickém pořadí (po kontrakci silně souvislých komponent vznikne acyklický graf).

2. Ohodnoť všechny literály z topologicky poslední komponenty souvislosti U na hodnotu 1. Z duality plyne, že $\bar{U}$ je topologicky první komponenta, ohodnocení U na 1 vynucuje ohodnocení $\bar{U}$ na 0. Pokud to není možné, znamená to, že $U = \bar{U}$ a splňující ohodnocení neexistuje (to ale nemůže nastat díky tomu, že jsme předpokládali, že $(\forall x \in V)$ jsou x a $\neg x$ v různých silně souvislých komponentách).

3. Odeber z G_F silně souvislé komponenty U a $\bar{U}$.

4. Dokud je G_F neprázdný, iteruj.

Uvedené ohodnocení (pokud existuje) jistě splní podmínky předchozího tvrzení. □

K otestování splnitelnosti formule tak stačí zkonstruovat graf G_F, najít jeho silně souvislé komponenty a zkontrolovat, zda pro žádné x neleží x a $\neg x$ v téže komponentě. Protože silně souvislé komponenty lze zkonstruovat v lineárním čase prohledáváním do hloubky ([17]), lze celé testování provést v lineárním čase.

Algoritmus 3 Algoritmus pro 2SAT založený na proceduře unitprop

Vstup: Kvadratická formule F v CNF na proměnných $x_1, x_2, \ldots, x_n$.
Výstup: Splnující pravdivostní ohodnocení F, nebo „nesplnitená“.

```
1: if F obsahuje prázdnou klauzuli then
2:   return „nesplnitelná“
3: end if
4: t := ∅
5: while F není prázdná množina klauzulí do
6:   v := libovolná proměnná vyskytující se v F
7:   (F′, t′) := unitprop( F ∪ {{¬v}} )
8:   if ∅ ∈ F′ then
9:     (F′, t′) := unitprop( F ∪ {{v}} )
10:    if ∅ ∈ F′ then
11:      return „nesplnitelná“
12:    end if
13:  end if
14:  F := F′
15:  t := t ∪ t′
16: end while
17: return t
```

Nyní slíbená poznámka o algoritmu založeném na proceduře unitprop. Korektnost tohoto algoritmu lze snadno nahlédnout. Stačí si uvědomit si jak se chová procedura unitprop vzhledem k implikačnímu grafu.

Jednotková klauzule l odpovídá v incidenčním grafu ohodnocení literálu l na 1 a $\neg l$ na 0. Toto ohodnocení vynutí ohodnocení všech literálů do kterých z l vede orientovaná cesta na 1 (a z duality všech literálů, ze kterých vede orientovaná cesta do $\neg l$ na 0). Právě tato vynucená ohodnocení provádí unitprop, protože hrany v incidenčním grafu jsou v korespondenci s ryze kvadratickými klauzulemi. Pakliže v grafu vede cesta z l do $\neg l$, odvodí unitprop prázdnou klauzuli. V tom případě se zkusí opačné přiřazení hodnot. Když ani to nevede k cíli, znamená to, že mezi l a $\neg l$ vede orientovaná cesta oběma směry a jsou tedy v jedné silně souvislé komponentě.

3.2 Balanced formule

Balanced formule (také CC-balanced) jsou aplikací teorie okolo balanced matic. Každá balanced matice určuje (společné se systémem nerovností popsaných níže) polytop jehož všechny vrcholy mají celočíselné souřadnice. Známé výsledky o nich včetně použití pro testování splnitelnosti jsou shrnuty v článku [6], odkud také pochází většina definic a výsledků zde uvedených.

Definice 3.2.1. [6] Matice A s prvky z $\{-1, 0, 1\}$ je *hole* (z angličtiny), jestliže A obsahuje právě dva nenulové prvky na každém řádku a v každém sloupci a žádná její vlastní podmatice nemá tuto vlastnost.

Pozorování 3.2.2. [6] *Každá hole matice je čtvercová.*

Důkaz. Nechť A je hole matice typu $m \times n$. Každý řádek i sloupec tedy obsahuje dvě nenulové hodnoty. Když sečteme všechny dvojky za každý řádek, dostaneme počet

nenulových prvků matice $(2n)$. Stejně pro sloupce číslo $2m$ vyjadřuje tentýž počet nenulových prvků. Je tedy $2m = 2n \Rightarrow m = n$ a matice je čtvercová. □

Poznámka 3.2.3. [6] Každá hole matice může být permutací sloupců a řádků přeuspořádána tak, že má nenulové prvky (± 1) pouze na hlavní diagonále $(h_{i,i})$, na pozicích nad hlavní diagonálou $(h_{i,i+1})$ a na pozici vlevo dole $(h_{n,1})$.

Součet prvků v každé hole matici velikosti alespoň 2 je sudý.

Definice 3.2.4. [6] Hole matice $A = (a_{i,j})$ je *sudá* [resp. *lichá*], jestliže součet jejích prvků $\sum_{i,j=1}^{n}(a_{i,j}) \equiv 0 \pmod 4$ [resp. $\sum_{i,j=1}^{n}(a_{i,j}) \equiv 2 \pmod 4$].

Matice A s prvky z $\{0, \pm 1\}$ je *balanced*, jestliže žádná její podmatice není lichá hole matice.

Formule F je *balanced*, jestliže její incidenční matice M_F (viz definice 2.3.4) je balanced.

Definice 3.2.5. [6] Nechť A je matice s prvky z $\{-1, 0, 1\}$. Nechť $n(\vec{A})$ je vektor jehož složky udávají počet hodnot -1 v jednotlivých řádcích matice A. Nechť $\vec{e}$ je vektor samých jedniček.

Definujme následující tři polytopy (vnořené do jednotkové hyperkrychle):
$P(A) = \{\vec{x} \in \mathbb{R}^n : A \cdot \vec{x} \leq \vec{e} - n(\vec{A}), \vec{0} \leq \vec{x} \leq \vec{e}\}$ (vnitřní),
$Q(A) = \{\vec{x} \in \mathbb{R}^n : A \cdot \vec{x} \geq \vec{e} - n(\vec{A}), \vec{0} \leq \vec{x} \leq \vec{e}\}$ (vnější) a
$R(A) = \{\vec{x} \in \mathbb{R}^n : A \cdot \vec{x} = \vec{e} - n(\vec{A}), \vec{0} \leq \vec{x} \leq \vec{e}\}$ (hraniční).

Říkáme, že polytop je *celočíselný*, jestliže všechny jeho vrcholy mají všechny složky celočíselné (v uvedených případech to znamená, že jsou z $\{0, 1\}$).

Naším záměrem bude dokázat, že uvedené polytopy $P(A)$, $Q(A)$ a $R(A)$ jsou celočíselné, právě když je matice A balanced (věta 3.2.9). Z toho pak bude možno nahlédnout, že splnitelnost lze vyřešit některou klasickou metodou lineárního programování. K důkazu budou třeba následující pomocná tvrzení.

Pozorování 3.2.6. [6] *Nechť A je hole $\{0, \pm 1\}$-matice. Jestliže A je sudá, pak je singulární (tj. $det(A) = 0$), jestliže je lichá, pak je determinant $det(A) = \pm 2$.*

Důkaz. Plyne přímo z výpočtu determinantu nad maticí upravenou permutací sloupců a řádků do tvaru uvedeného v poznámce 3.2.3. □

Lemma 3.2.7. [6] *Jestliže A je balanced $\{0, \pm 1\}$-matice, pak polytop $R(A)$ je celočíselný.*

Důkaz. Pro spor předpokládejme, že balanced $\{0, \pm 1\}$-matice A je nejmenší (součtem počtu řádků a sloupců) taková, že polytop $R(A)$ není celočíselný. To znamená, že $R(A)$ je neprázdný (z toho, že má vrchol).

Buď $\vec{x'}$ neceločíselný vrchol $R(A)$. Protože A je nejmenší možná, platí $(\forall j \in \{1, \ldots, n\}) 0 < x'_j < 1$ (jinak bychom mohli pomocí projekce získat polytop v méněrozměrném prostoru – s menší maticí A). Z toho také plyne, že A je čtvercová regulární a z toho pak že $\vec{x'}$ je jednoznačně určen (jediný vektor v $R(A)$).

Označme $a^1, \ldots, a^n$ řádky matice A a A_i matici vzniklou z A vypuštěním i-tého řádku. Z minimality A určuje A_i celočíselný polytop $R(A_i)$. Protože A je čtvercová regulární, má $R(A_i)$ právě dva vrcholy. Označme je $\vec{x_s}$ a $\vec{x_t}$. Z toho, že $\vec{x'}$ je v $R(A_i)$, plyne, že $\vec{x'} = \lambda\vec{x_s} + (1 - \lambda)\vec{x_t}$ pro $\lambda \in (0, 1)$. Vzhledem k tomu, že

$(\forall j \in \{1,\ldots,n\})0 < x'_j < 1$ a $\vec{x_s}$ a $\vec{x_t}$ mají všechny složky 0 nebo 1, je $\vec{x_s} + \vec{x_t} = \vec{e}$ (máme vektor $\vec{x'}$ ostře mezi vektory se souřadnicemi 0/1).

Buď k libovolný řádek A_i. Protože $\vec{x_s}$ a $\vec{x_t}$ splňují podmínky $a^k \cdot \vec{x} = 1 - n(a^k)$, platí také $a^k \cdot \vec{e} = 2(1 - n(a^k))$, tzn. řádek k obsahuje právě dvě nenuly. Použijeme-li tento argument na všechny matice A_i, dostáváme, že každý řádek původní matice A obsahuje právě dvě nenuly.

Jestliže A má sloupec j s pouze jednou nenulou $a_{k,j}$, odstraňme z A sloupec j a řádek k. Vzhledem k tomu, že A byla regulární, je vzniklá matice opět regulární a absolutní hodnota determinantu se nezměnila (odebrali jsme hodnotu ± 1 jedinou v daném sloupci). Opakováním tohoto postupu můžeme získat regulární čtvercovou matici, která má dvě nenulové hodnoty v každém řádku a každém sloupci (označme ji B). B může být přeuspořádána do blokového diagonálního tvaru takového, že všechny podmatice jsou hole. Protože B je regulární, jsou regulární i všechny tyto podmatice a z pozorování 3.2.6 jsou všechny liché. To znamená, že A není balanced, což je spor. □

Věta 3.2.8. [6] *Nechť A je balanced $\{0, \pm 1\}$-matice s řádky $a^i, i \in S$. Nechť S_1, S_2 a S_3 tvoří rozklad S. Potom*

$$T(A) = \{x \in \mathbb{R}^n : \begin{array}{ll} a^i\vec{x} \geq 1 - n(a^i) & \textit{pro } i \in S_1, \\ a^i\vec{x} = 1 - n(a^i) & \textit{pro } i \in S_2, \\ a^i\vec{x} \leq 1 - n(a^i) & \textit{pro } i \in S_3, \\ \vec{0} \leq \vec{x} \leq \vec{e}\} & \end{array}$$

je celočíselný polytop.

Důkaz. Jestliže x' je vrchol $T(A)$, je to také vrchol polytopu, který vznikne odstraněním všech nerovností, pro něž s x' nenastává rovnost (odstranění nadrovin, které neprocházejí vrcholem x'). Z předchozího lemmatu plyne, že každý vrchol tohoto polytopu má složky z $\{0, 1\}$. □

Věta 3.2.9. [6] *Nechť A je $\{0, \pm 1\}$-matice, potom jsou následující tvrzení ekvivalentní:*

a) A je balanced,

b) $P(A)$ je celočíselný polytop.

c) $Q(A)$ je celočíselný polytop.

d) $R(A)$ je celočíselný polytop.

Důkaz. Protože balanced matice jsou uzavřeny na získávání podmatic, je dle předchozí věty $P(A)$, $Q(A)$ i $R(A)$ celočíselný polytop.

Předpokládejme, že A obsahuje lichou hole podmatici H. Dle pozorování 3.2.6 je $\vec{x} = (\frac{1}{2}, \ldots, \frac{1}{2})$ jednoznačně určené řešení rovnice $H\vec{x} = \vec{e}$, z čehož plynou opačné implikace. □

Jak najít splňující ohodnocení formule, jejíž incidenční matice M_F je balanced, je zřejmé. Stačí najít libovolné řešení uvnitř polytopu $P(M_F)$ — otestovat polytop na neprázdnost. To, že je formule balanced a polytop $P(M_F)$ neprázdný, znamená, že jsou všechny vrcholy tohoto polytopu celočíselné (a řešením). Nalezený vektor je

pak interpretován tak, že jeho i-tá složka odpovídá pravdivostnímu ohodnocení i-té proměnné (hodnota 0 odpovídá nepravdě a hodnota 1 pravdě).

K řešení úlohy stačí použít algoritmus pro lineární programování s reálnými čísly. Ty navíc hledají pouze mezi vrcholy polytopu (je-li omezený, což zde je díky podmínkám $0 \leq x_i \leq 1$), takže vrátí přímo řešení.

V citovaném textu [6] je navíc ukázáno, že lze použít algoritmus pracující efektivněji než obecné lineární programování. Z tohoto algoritmu navíc přímo vyplyne, že balanced formule jsou podtřídou třídy SLUR, což budeme potřebovat později. K tomu, aby bylo zřejmé, že tento algoritmus je korektní, dokážeme následující tvrzení.

Tvrzení 3.2.10. [6] *Nechť F je balanced formule, jejíž každá klauzule obsahuje alespoň dva literály. Pak pro každou proměnnou x_i existují splňující ohodnocení v_0 a v_1 takové, že platí $v_0(x_i) = 0$ a $v_1(x_i) = 1$.*

Důkaz. Protože vektor $\vec{x} = (\frac{1}{2}, \ldots, \frac{1}{2})$ patří do polytopu $Q(M_F)$ a ten je celočíselný, může být $\vec{x}$ vyjádřen jako konvexní kombinace některých vrcholů tohoto polytopu. Zřejmě pro každé i existují dva vektory této kombinace, kde jeden má $x_j = 0$ a druhý $x_j = 1$ (argument obdobný tomu, jež se vyskytl v důkazu lemmatu 3.2.7). □

Algoritmus 4 Algoritmus pro Balanced formule [6]

Vstup: Formule F v CNF na proměnných $x_1, x_2, \ldots, x_n$.
Výstup: Splnující ohodnocení proměnných z F, nebo „nesplnitená“.

```
(F, t) := unitprop(F)
if F obsahuje prázdnou klauzuli then
    return „nesplnitelná“
end if
while F není prázdná množina klauzulí do
    v := libovolná proměnná vyskytující se v F
    Proveď libovolný z následujících kroků:
    1. (F, t') := unitprop( F ∪ {{¬v}} )
    2. (F, t') := unitprop( F ∪ {{v}} )
    if ∅ ∈ F then
        return „nesplnitelná“
    end if
    t := t ∪ t'
end while
return t
```

Tvrzení 3.2.11. *Třída Balanced formulí je v inkluzi neporovnatelná se třídou kvadratických formulí.*

Důkaz. Kvadratické $\not\subseteq$ Balanced: Formule $F = (x_1 \vee x_2)\&(x_1 \vee \neg x_2)$ je kvadratická, ale není ve třídě Balanced. Odpovídá jí totiž matice

$$M_F = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

a tato matice není balanced, protože má součet svých prvků roven 2, je tak sama o sobě lichá hole matice.

Balanced $\not\subseteq$ Kvadratické: Formule $(x_1 \vee x_2 \vee x_3)$ zjevně není kvadratická. Její incidenční matice $M_F = (\ 1 \quad 1 \quad 1\ )$ zjevně neobsahuje lichou hole podmatici (čtvercovou matici se dvěma nenulovými prvky v každém řádku a sloupci), je tedy balanced. □

Tvrzení 3.2.12. *Třída balanced formulí není uzavřená na komplementaci literálu, spojení formulí, odebrání literálu. Je uzavřená na komplementaci proměnné, odebrání proměnné a klauzule, částečné dosazení.*

Důkaz. K důkazu neuzavřenosti na komplementaci literálu vezměme formuli $(x_1 \vee x_2)\&(\neg x_1 \vee \neg x_2)$. Ta má matici

$$M_F = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix},$$

která je sudá hole. Komplementací literálu $\neg x_1$ získáme formuli $(x_1 \vee x_2)\&(x_1 \vee \neg x_2)$, která není Balanced, jak se praví v důkazu tvrzení 3.2.11.

K důkazu neuzavřenosti na spojení dvou formulí můžeme použít $F_1 = (x_1 \vee x_2)$ a $F_2 = (x_1 \vee \neg x_2)$. Ty jsou balanced a jejich spojením získáme formuli z předchozího odstavce.

Formule $F = (x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ je balanced, protože má matici $M_F = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{pmatrix}$, a ta neobsahuje jicho hole podmatici. Když formuli spojíme čtyřikrát za sebe (a řádky matice čtyřikrát pod sebe), získáme formuli, která je opět balanced. Postupným odebíráním literálů pak můžeme vytvořit formuli $F' = (x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)\&(x_1 \vee x_2)\&(\neg x_1 \vee \neg x_2)\&(x_2 \vee x_3)\&(\neg x_2 \vee \neg x_3)\&(x_1 \vee x_3)\&(\neg x_1 \vee \neg x_3)$, která již není balanced, jak dokazuje tučně vyznačená lichá hole podmatice v její incidenční matici:

$$M_{F'} = \begin{pmatrix} 1 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \\ -1 & -1 & -1 \\ -1 & 0 & -1 \\ 0 & -1 & -1 \\ -1 & -1 & 0 \end{pmatrix}.$$

Uzavřenost na komplementaci proměnné dokážeme sporem. Nechť F' vznikne z F komplementací proměnné x a nechť $M_{F'}$ obsahuje lichou hole podmatici. Každá hole matice má v každém sloupci právě dvě nenulové hodnoty ± 1. Komplementací proměnné x těmto hodnotám v příslušném sloupci změníme znaménko, což se do celkového součtu projeví hodnotou, která je násobkem 4 (byl-li součet ve sloupci 0, je po změně znaménka opět 0; byl-li ± 2, je po změně znaménka ∓ 2). To znamená, že se nezmění sudost/lichost matice a podmatice stejného typu tak musela být i v M_F – což je spor.

Odebrání proměnné v odpovídá odebrání sloupce matice M_F („přečíslováním“ proměnných můžeme zajistit, že to bude poslední sloupec). Tím ale nemůže vzniknout nová lichá hole podmatice: Kdyby v matici po odstranění sloupce byla lichá hole podmatice, pak musela být stejná podmatice i v M_F.

K uzavřenosti na odebrání klauzule lze užít stejnou úvahu s tím rozdílem, že mažeme řádek matice M_F namísto sloupce.

Uzavřenost třídy na částečné dosazení plyne z toho, že je uzavřená na odebrání proměnné i klauzule. □

Jako poslední neprodiskutovaná otázka zbývá rozpoznávání třídy Balanced. Žádný ze známých algoritmů na rozpoznávání pracujících v polynomiálním čase není přímočarý, proto zájemce odkazujeme na článek [6].

3.3 Matched formule

Třída Matched formulí byla definována v článku [11].

Definice 3.3.1. Nechť F je formule nad proměnnými V. *Incidenční graf* pro formuli F je bipartitní graf, kde jednu partitu tvoří vrcholy odpovídající klauzulím a druhou vrcholy odpovídající všem proměnným z V. Hrana vede mezi dvěma vrcholy právě, když se proměnná vyskytuje v dané klauzuli.

Definice 3.3.2. Nechť $G = (U_G \cup V_G, E_G)$ je bipartitní graf s partitami U_G a V_G. $E_P \subseteq E_G$ nazveme *totálním párováním ze strany* U_G, jestliže do každého vrcholu U_G vede právě jedna hrana E_P a do každého vrcholu V_G vede nejvýše jedna hrana E_P.

Definice 3.3.3. Formule patří do třídy Matched, jestliže v incidenčním grafi existuje totální párování ze strany partitity klauzulí.

Pozorování 3.3.4. *Každá formule ze třídy Matched je splnitelná.*

Důkaz. Je-li formule ve třídě Matched, má v incidenčním grafu totální párování ze strany klauzulí. Z tohoto párování vytvoříme splňující ohodnocení: Pro každou klauzuli vezmeme tu proměnnou, do které vede hrana v totálním párování a nastavíme její hodnotu tak, aby byla klauzule splněna. Díky tomu, že je párování totální, nehrozí, že bychom chtěli změnit hodnotu již ohodnocené proměnné (každá klauzule má vlastní proměnnou, která jí splní). □

Poznámka 3.3.5. Jestliže formule F je ve třídě matched, n je počet jejích proměnných a m počet klauzulí. Pak je $m \leq n$.

Tvrzení 3.3.6. *Třída Matched je uzavřená na komplementaci literálu (a tedy i proměnné) a na odebrání klauzule. Není uzavřená na odebrání literálu, částečné dosazení a spojení dvou formulí.*

Důkaz. V případě, že provedeme komplementaci literálu, mohou nastat dva případy. Změní-li se literál klauzule, který není pro tuto klauzuli v totálním párování, nemá to vůbec žádný vliv na párování ani splňující ohodnocení. Jestliže se změní párovací literál, zůstává párování stále použitelné (protože se jedná o graf „klauzule-proměnné“), jen se při sestavování splňujících ohodnocení přiřadí opačná hodnota.

Odebrání klauzule se projeví odebráním příslušného vrcholu incidenčního grafu a hrany v totálním párování (to zůstane totálním párováním pro vzniklou formuli).

Jako protipříklad na neuzavřenost třídy Matched na odebrání literálu můžeme použít formuli $(x_1 \vee x_2)\&(\neg x_2)$. Ta je Matched díky totálnímu párování vybírajícímu

z první klauzule proměnnou x_1 a z druhé x_2. Jestliže odebereme literál x_1, získáme nesplnitelnou formuli, která není Matched.

Neuzavřenost na částečné dosazení je dána tím, že ze splnitelné matched formule můžeme nevhodným dosazením získat nesplnitelnou formuli. Například dosadíme-li v předchozí formuli za x_1 hodnotu 0.

Neuzavřenost na spojení dvou formulí ukazuje triviální příklad $F_1 = x$ a $F_2 = \neg x$. Pak $F = F_1 \& F_2$ jistě není matched. □

Jelikož je každá formule třídy Matched splnitelná, algoritmus na testování splnitelnosti nemá smysl uvádět.

Naopak je třeba uvést, jak najít totální párování (případně rozhodnout o jeho neexistenci). To lze udělat několika způsoby. Například pomocí některého algoritmu na toky v sítích. Incidenční graf zorientujeme tak, že hrany vedou z vrcholů partity klauzulí do vrcholů v partitě proměnných. Přidáme zdroj – vrchol ze kterého vedou hrany do všech vrcholů reprezentujících klauzule a stok za všechny vrcholy odpovídající proměnným. Váhy všech hran nastavíme na hodnotu 1. Najdeme (celočíselný) tok v této síti. Totální párování ze strany klauzulí pak odpovídá hranám, přes které teče jednotkový tok. Jestliže existuje vrchol odpovídající klauzuli, přes který nic neteče (tzn. celkový tok $<$ počet klauzulí), totální párování neexistuje.

Protože existují polynomiální algoritmy na hledání toků v sítích, lze i totální párování najít v polynomiálním čase.

Tvrzení 3.3.7. *Třída Matched je v inkluzi neporovnatelná s třídou kvadratických formulí i třídou Balanced.*

Důkaz. Matched $\not\subseteq$ Kvadratické: například formule $(x_1 \vee x_2 \vee x_3)$ není kvadratická a je matched (se třemi totálními párováními).

Kvadratické $\not\subseteq$ Matched: $(x_1 \vee x_2)\&(\neg x_1 \vee x_2)\&(x_1 \vee \neg x_2)$ je kvadratická a není matched, protože obsahuje více klauzulí než proměnných.

Matched $\not\subseteq$ Balanced: Formule $F = (x_1 \vee x_2)\&(x_1 \vee \neg x_2)$ je Matched (párování vybere např. z první klauzule x_1 a z druhé x_2), ale není Balanced (jak bylo ukázáno v tvrzení 3.2.11).

Balanced $\not\subseteq$ Matched: zde stačí uvést libovolnou nesplnitelnou balanced formuli (má prázdný polytop $P(M_F)$ a pro ten platí, že všechny jeho vrcholy jsou celočíselné, protože žádné nemá). Takovou formulí je třeba $x_1 \& \neg x_1$. □

3.4 Hornovské a skrytě hornovské formule

Definice 3.4.1. Formule F je *hornovská*, jestliže každá její klauzule obsahuje nejvýše jeden pozitivní literál.

Tvrzení 3.4.2. *Třída hornovských formulí je neporovnatelná se třídami kvadratických, balanced a matched formulí.*

Důkaz. Hornovské $\not\subseteq$ Kvadratické: $(x_1 \vee x_2 \vee x_3)$ je hornovská, ale ne kvadratická.

Kvadratické $\not\subseteq$ Hornovské: $(x_1 \vee x_2)\&(\neg x_1 \vee x_2)\&(x_1 \vee \neg x_2)\&(\neg x_1 \vee \neg x_2)$ je kvadratická, ale není hornovská.

Hornovské $\not\subseteq$ Balanced: $(\neg x_1 \vee \neg x_2)\&(\neg x_1 \vee x_2)$ jde vlastně o formuli použitou v tvrzení 3.2.11 po komplementaci obou proměnných. Vzhledem k tomu, že třída

Balanced je na tuto operaci uzavřená, nemůže být balanced ani tato formule. Na druhou stranu zjevně partří do třídy hornovských formulí.

Balanced $\not\subseteq$ Horn: $F = (x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ je balanced, protože jí odpovídá matice $M_F = \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{pmatrix}$, která neobsahuje lichou hole podmatici. Formule F není hornovská, protože její první klauzule obsahuje více než jeden pozitivní literál.

Hornovské $\not\subseteq$ Matched: $x_1\&\neg x_1$ je hornovská, ale není Matched. Je totiž nesplnitelná.

Matched $\not\subseteq$ Horn: $(x_1 \vee x_2 \vee \neg a)\&(\neg x_1 \vee x_2 \vee \neg b)\&(x_1 \vee \neg x_2 \vee \neg c)\&(\neg x_1 \vee \neg x_2 \vee \neg d)$ je ze třídy Matched, protože v každé klauzuli je literál, který zajistí existenci párování (např. a, b, c a d). Nejedná se však o hornovskou formuli, protože její první klauzule obsahuje dva pozitivní literály. □

Pozorování 3.4.3. *Třída hornovských formulí je uzavřená na odebrání literálu i klauzule, částečné dosazení a spojení dvou formulí. Není uzavřená na komplementaci literálu ani proměnné.*

Důkaz. Žádná z operací, na které je třída hornovských formulí uzavřená, nepřidává pozitivní literál do klauzule.

Formule ukazující, že není uzavřená na komplementaci literálu a proměnné je triviální: $(x_1 \vee \neg x_2)$ je hornovská, ale pokud komplementujeme $\neg x_2$, získáme formuli $(x_1 \vee x_2)$, která hornovská není. □

Třídu hornovských formulí lze rozpoznávat triviálním způsobem — projít formuli a zkontrolovat, že každá klauzule má nejvýše jeden pozitivní literál. Toto rozpoznávání je zřejmě lineární v délce formule.

Pro testování splnitelnosti hornovských formulí uvedeme algoritmus lineární v délce formule.

Pozorování 3.4.4. *Jestliže formule neobsahuje jednotkovou klauzuli s pozitivním literálem, pak je odhonocení přiřazující každé proměnné hodnotu 0 splňující.*

Důkaz. Dle předpokladu může formule obsahovat pouze negativní jednotkové klauzule (ty jsou splněny dosazením nuly) a klauzule delší (ty mohou obsahovat nejvýše jeden pozitivní literál, obsahují tedy alespoň jeden negativní a ten je nulovým ohodnocením splněn). □

Algoritmus 5 se snaží částečným dosazováním za pozitivní literály vytvořit formuli bez jednotkových klauzulí s pozitivním literálem. Pro zbylé proměnné vezme částečné dosazení přiřazující všem hodnotu 0 (a tím jí splní).

Tento postup je korektní, protože když za pozitivní klauzuli dosadíme hodnotu 1, klauzule zmizí jako splněná.

Algortimus lze implementovat lineárně v délce formule. Stačí formuli reprezentovat pomocí struktury uvedené v části 2.2 na obrázku 2.1 a uchovávat si spojový seznam pozitivních literálů.

Ve skutečnosti lze algoritmus chápat jako zavolání procedury unitprop a dosazení nuly za proměnné, které ve formuli zbydou.

Definice 3.4.5. Formule je *skrytě hornovská*, jestliže z ní lze komplementací některých proměnných získat hornovskou formuli.

Algoritmus 5 Algoritmus pro hornovské formule

Vstup: Hornovská formule F v CNF na proměnných $x_1, x_2, \ldots, x_n$.

Výstup: Splnující částečné pravdivostní ohodnocení proměnných z F, nebo „nesplnitená“.

1: $t := \{(x_i, 0) : i \in \{1, \ldots, n\}\}$
2: **while** (F obsahuje pozitivní literál jako jednotkovou klauzuli) & (F neobsahuje prázdnou klauzuli) **do**
3: $x :=$ libovolná proměnná vyskytující se v F pozitivně jako jednotková klauzule
4: $t := (t \setminus \{(x, 0)\}) \cup \{(x, 1)\}$
5: $F := \{C \setminus \{\neg x\} : C \in F \ \& \ x \notin C\}$
6: **end while**
7: **if** F obsahuje prázdnou klauzuli **then**
8: **return** „nesplnitelná“
9: **else**
10: **return** t
11: **end if**

Uvedeme lineární algoritmus (pocházející z [2]) na hledání množiny proměnných, jejichž komplementací ze vstupní formule vznikne hornovská formule. Pokud taková množina neexistuje, algoritmus to oznámí.

Algoritmus lze použít k rozpoznávání skrytě hornovských formulí. Společně s výše uvedeným algoritmem také k testování splnitelnosti (převedeme formuli na hornovskou a použijeme dříve uvedený algoritmus).

Poznámka 3.4.6. Pokud chceme převést skrytě hornovskou formuli na hornovskou, nemusíme se starat o lineární termy. Ty totiž nemají vliv na to, zda je formule hornovská nebo není.

Definice 3.4.7. [2] Nechť F je formule v CNF. Definujeme k ní formuli F_q následujícím způsobem. F_q obsahuje klauzuli $(l_1 \vee l_2)$, právě když F obsahuje klauzuli, v níž se vyskytují literály l_1 a l_2.

Pozorování 3.4.8. *Délka formule F_q je $\leq$ druhá mocnina délky F.*

Důkaz. Jedná se o jednoduchou úvahu s kombinačními čísly. Počet různých dvojic nad k prvkovou množinou je $\binom{k}{2} = \frac{1}{2}k^2 - \frac{1}{2}k$. □

Lemma 3.4.9. [2] *Formule F je hornovská, právě když ohodnocení přiřazující všem proměnným 0 splní F_q.*

Důkaz. Nechť nejprve F je hornovská. To znamená, že každá klauzule obsahuje nejvýše jeden pozitivní literál. Z toho plyne, že každá klauzule F_q obsahuje nejvýše jeden pozitivní literál. Protože je F_q ryze kvadratická, obsahuje každá její klauzule alespoň jeden negativní literál. Ten je splněn ohodnocením příslušné proměnné na 0. Ohodnocení všech proměnných nulou je tedy pro F_q splňující.

Opačnou implikaci dokážeme sporem. Nechť F_q je splněna ohodnocením samými nulami a nechť F není hornovská. To, že F není hornovská, znamená, že obsahuje klauzuli s alespoň dvěma pozitivními literály. Ta se v F_q projeví jako klauzule C s oběma literály pozitivními. C ale není splněna ohodnocením přiřazujícím všem proměnným hodnotu 0, což je spor. □

Věta 3.4.10. [2] *Formule F je skrytě hornovská, právě když je formule F_q splnitelná. Navíc je-li v splňující ohodnocení pro F_q, pak komplementací těch proměnných x, pro které je $v(x) = 1$, získáme z formule F hornovskou formuli F'.*

Důkaz. Nechť F je skrytě hornovská formule a nechť S je množina proměnných taková, že po komplementaci proměnných množiny S vznikne z formule F hornovská formule F' (a jí odpovídající formule ${F_q}'$). Podle předchozího lemmatu je ${F_q}'$ splněna ohodnocením samými nulami. Z tohoto ohodnocení vytvoříme negací hodnot proměnných z S ohodnocení

$$v(x) = \begin{cases} 0, & x \notin S \\ 1, & x \in S \end{cases}$$

které splní F_q.

Nyní ukážeme opačnou implikaci. Nechť F_q je splnitelná a nechť v je ohodnocení splňující F_q. Pro toto ohodnocení definujeme množinu $S = \{x : v(x) = 1\}$. Nechť nyní ${F_q}'$ vznikne z F_q komplementací proměnných z množiny S. Buď $C = (p \vee q)$ libovolná klauzule F_q. Protože v splňuje F_q máme $\bar{v}(p) = 1$ nebo $\bar{v}(q) = 1$ (bez újmy na obecnosti předpokládejme $\bar{v}(p) = 1$). Nyní rozebereme dva případy podle toho, zda je p přímý, nebo negovaný literál:

a) $p = x$. Pak $v(x) = 1$. Tím pádem $x \in S$, což znamená, že v odpovídající klauzuli ${F_q}'$ se vyskytuje literál $\neg x$.

b) $p = \neg x$. Pak $v(x) = 0$. To znamená, že $x \notin S$ a tedy se $\neg x$ vyskytuje také v odpovídající klauzuli ${F_q}'$.

${F_q}'$ je tak splněna ohodnocením samými nulami. Použitím lemmatu pak dostáváme, že formule F' je hornovská. Z toho pak plyne, že F je skrytě hornovská. □

Výše uvedená tvrzení naznačují, jak rozpoznat a převést skrytě hornovskou formuli na hornovskou v kvadratickém čase (délka kvadratické formule F_q je omezena druhou mocninou délky F a na splnitelnost kvadratické formule máme lineární algoritmus). Dokážeme ještě tvrzení, které nám dá návod na lineární algoritmus.

Definice 3.4.11. [2] Nechť F je formule v CNF. Definujeme k ní formuli F_p následujícím způsobem: pro každou její klauzuli $C = (l_1 \vee \ldots \vee l_k)$ přidáme do F_p následující výraz jako podformuli

$$(l_1 \vee {y_1}^C) \& \bigwedge_{1<j<k} \left((\neg {y_{j-1}}^C \vee l_j) \& (\neg {y_{j-1}}^C \vee {y_j}^C) \& (l_j \vee {y_j}^C) \right) \& (\neg {y_{k-1}}^C \vee l_k)$$

Pozorování 3.4.12. *Délka F_p je lineární vzhledem k délce F.*

Důkaz. Za jednotkovou klauzuli budou v F_p dva literály, za kvadratickou čtyři. Klauzuli o délce $k \geq 3$ zastupuje výraz o délce $6(k-2)+4 = 6k-8$ literálů. To znamená ve všech případech nejvýše lineární nárůst. Platí tedy $|F_p| = O(|F|)$. □

Věta 3.4.13. [2] *Nechť F je formule v CNF a F_p, F_q jsou formule uvedeného tvaru pro formuli F. Pak F_p je splnitelná právě když je splnitelná F_q.*

Důkaz. Buď v pravdivostní ohodnocení, pro které je $\bar{v}(F_p) = 1$. Nechť $(l_i \vee l_j)$ je libovolná klauzule z F_q. Pak v F existuje kluazule $C = (l_1 \vee \ldots \vee l_i \vee \ldots \vee l_j \vee \ldots \vee l_k)$. Tato klauzule C vygeneruje do F_p mimo jiné klauzule $(l_i \vee y_i^C)$, $(\neg y_i^C \vee y_{i+1}^C)$, $\ldots$, $(\neg y_{j-1}^C \vee l_j)$. Platí-li $\bar{v}(F_p) = 1$, pak nemůže být $\bar{v}(l_i) = \bar{v}(l_j) = 0$. To znamená že i $\bar{v}(l_i \vee l_j) = 1$, a tudíž že $\bar{v}(F_q) = 1$.

Pro opačnou implikaci ukážeme, že libovolné splňující ohodnocení v formule F_q lze rozšířit na splňující ohodnocení F_p. Pro klauzuli $C = (l_1 \vee \ldots \vee l_k)$ z původní formule F máme v F_q jako podformuli $Q = \bigwedge_{1 \leq i < j \leq k} (l_i \vee l_j)$. Protože v je splňující ohodnocení F_q, tak pro nejvýše jeden literál l_s platí $\bar{v}(l_s) = 0$. Definujeme ohodnocení v' rozšiřující v následujícím způsobem:

$$v'(x) = \begin{cases} 0, & \text{pro } y_i^C, 1 \leq i < s \\ 1, & \text{pro } y_i^C, s \leq i \leq k \\ v(x), & \text{pro proměnné formule } F \end{cases}$$

Provedeme-li toto rozšíření ohodnocení pro všechny klauzule formule C, získáme splňující ohodnocení pro F_p. To lze nahlédnout snadněji, představíme-li si tu část formule F_q zastupující C jako řetězec implikací (tak jako v části o kvadratických formulích — hodnota 0 zajišťuje splnění části vlevo od s a hodnota 1 části vpravo). □

Celý postup pro rozpoznávání a řešení skrytě hornovských formulí je následující:

1. Z formule F zkonstruujeme F_p.
2. Pomocí algoritmu pro kvadratické formule rozhodneme o splnitelnosti F_p.
3. Jestliže F_p není splnitelná, F není skrytě hornovská. Jestliže naopak v splňuje F_p, získáme z F hornovskou F' komplementací těch proměnných x formule F, pro které je $v(x) = 1$.

Pozorování 3.4.14. *Třída skrytě hornovských formulí je uzavřená na odebrání literálu i klauzule, částečné dosazení, komplementací proměnné, spojení dvou formulí, Není uzavřená na komplementací literálu.*

Důkaz. K důkazu neuzavřenosti na komplementaci literálu lze užít formule $(x_1 \vee \neg x_2 \vee \neg a) \& (x_1 \vee \neg x_2) \& (\neg x_1 \vee x_2) \& (\neg x_1 \vee \neg x_2)$. Ta je hornovská, tedy i skrytě hornovská. Komplementací prvního výskytu literálu $\neg x_2$ získáme formuli $(x_1 \vee x_2 \vee \neg a) \& (x_1 \vee \neg x_2) \& (\neg x_1 \vee x_2) \& (\neg x_1 \vee \neg x_2)$, která až není skrytě hornovská. To proto, že se v ní vyskytují všechny kombinace přímosti a negovanosti proměnných x_1 a x_2. Komplementací proměnné tedy nezamezíme výskytu $(x_1 \vee x_2)$ (pozn. literál $\neg a$ se ve formuli vyskytuje pouze z toho důvodu, aby neobsahovala dvě stejné klauzule).

Uzavřenost na komplementaci proměnné plyne přímo z definice třídy. Pro ostatní operace lze užít shodných argumentů, jako pro třídu hornovských formulí. □

Tvrzení 3.4.15. *Třída skrytě hornovských formulí obsahuje hornovské formule jako vlastní podtřídu. Je v inkluzi neporovnatelná se třídami kvadratických, balanced a matched formulí.*

Důkaz. To, že hornovské formule jsou podtřídou skrytě hornovských je jasné — stačí vzít prázdnou množinu proměnných ke komplementaci. Ostrost inkluze lze ukázat triviálně například formulí $(x_1 \vee x_2)$, která není hornovská, ale po komplementaci libovolného jejího literálu se hornovskou stane.

Pro neporovnatelnost uvedených tříd lze použít formule uvedené v tvrzení 3.4.2. □

3.5 Třída hidden extended horn

Za vznikem třídy stojí otázka, kdy lze celočíselné řešení úlohy lineárního programování získat zaokrouhlením řešení reálného problému se shodným systémem nerovností. Částečnou odpověď (postačující podmínky) na tuto otázku poskytl Chandrasekaran v článku [12]. Na základě jeho výsledků byly v [13] definovány třídy extended horn a hidden extended horn. Nyní uvedeme větu, na které je třída založena, již ve variantě pro formule.

Věta 3.5.1. [13] *Nechť M_F je incidenční matice pro formuli F. Uvažujme systém nerovností $M_F \cdot \vec{x} \geq \vec{e} - n(\vec{M_F})$, $\vec{x} \geq \vec{0}$, $-\vec{x} \geq -\vec{e}$, kde $\vec{e}$ je jednotkový vektor a $n(\vec{M_F})$ je vektor udávající počet -1 v řádcích matice M_F. Nechť dále T je regulární matice typu $n \times n$ splňující následující podmínky:*

a) T i T^{-1} jsou celočíselné matice,

b) Každý řádek T^{-1} obsahuje nejvýše jednu 1 a nejvýše jednu -1, žádné jiné nenulové hodnoty,

c) Každý řádek $M_F T^{-1}$ obsahuje nejvýše jednu zápornou hodnotu a to -1.

Potom pokud $\vec{x}$ splňuje podmínky uvedeného lineárního systému, splňuje je i celočíslený vektor $T^{-1}\lceil T\vec{x} \rceil$ (kde $\lceil T\vec{x} \rceil$ značí horní celou část $T\vec{x}$ po složkách). □

Uvedený systém nerovností odpovídá úloze lineárního programování z poznámky 2.3.6. V článku [13] byla ukázána grafová interpretace této věty (v případě incidenční matice formule). To vedlo k definici třídy extended horn, jíž po vysvětlení přechodu ke grafům také uvedeme.

T^{-1} musí díky své regularitě obsahovat v každém řádku alespoň jednu nenulovou hodnotu. Dle podmínky b) obsahuje nejvýše dvě nenulové hodnoty (-1 a 1). Můžeme tak přidáním sloupce s hodnotami z $\{0, \pm 1\}$ zajistit, že každý řádek obsahuje právě jednu -1 a jednu 1. Takto vzniklou matici lze interpretovat jako orientovaný graf G_T — sloupce odpovídají vrcholům, řádky hranám a hrana vede z vrcholu s hodnotou 1 do vrcholu s -1. Graf G_T má n hran a $n+1$ vrcholů (T^{-1} je typu $n \times n$ a přidali jsme k ní sloupec). Protože T^{-1} je regulární, má i matice s přidaným sloupcem lineárně nezávislé řádky a tedy neobsahuje cyklus (součet řádků odpovídajících hranám cyklu je $\vec{0}$; vynásobení řádku -1 odpovídá obrácení hrany, proto se v grafu nenachází ani neorientovaný cyklus). G_T je tudíž strom. Vrchol odpovídající poslednímu sloupci budeme označovat jako kořen (pozn. neexistuje vztah orientace hran a pozice kořene).

Dále pak můžeme řádky M_F chápat jako toky v grafu G_T. Hodnota 1 bude znamenat, že tok je ve směru hrany, hodnota -1 proti směru hrany, 0 nulový tok. Součinem $M_F T^{-1}$ získáme velikost toku, kterou jednotlivé vrcholy dodávají do sítě — rozdíl toho co z vrcholu odteče a co přiteče. V této interpretaci podmínka c) odpovídá požadavku, na existenci nejvýše jednoho vrchol (jiného než kořen) dodávajícího do sítě záporný tok – a ten musí být -1.

Definice 3.5.2. *Rozšířenou hvězdou* nazveme orientovaný strom, ve kterém všechny maximální orientované cesty vedou z jediného vrcholu (kořene).

Řekneme, že matice M_F *má rozšířenou hvězdu* G_T, jestliže každý tok odpovídající řádku matice M_F lze složit z:

i) jednotkového toku do kořene stromu G_T na všech hranách nějaké rozšířené hvězdy (může být i prázdná) jež je podstromem G_T,

ii) jednotkového souhlasně zorientovaného toku po nějaké neorientované cestě (i prázdné) obsažené v G_T.

Z uvedené věty plyne, že pokud má matice M_F vlastnosti rozšířené hvězdy pro nějaký strom G_T, pak lze celočíselné řešení (splňující ohodnocení) získat zaokrouhlením řešení reálného řešení. Strom G_T obecně neznáme. Pokud bychom ho znali, můžeme z něj zkonstruovat matici T^{-1}.

Třídy extended horn a hidden extended horn definujeme následujícím způsobem.

Definice 3.5.3. Formule F je *extended Horn*, jestliže její incidenční matice má nějakou rozšířenou hvězdu G_T.

Formule F je *hidden extended Horn*, jestliže její incidenční matice má rozšířenou hvězdu pro nějaký orientovaný zakořeněný strom (otáčením jeho hran — násobením řádků T^{-1} hodnotou -1 — lze získat hvězdu).

Pozorování 3.5.4. *Třída hidden extended horn je nadtřídou třídy hidden horn.*

Důkaz. Pro hornovské formule vezmeme jako matici T matici $-I$ (kde I je jednotková matice). Tato matice odpovídá hvězdě, kde jsou všechny hrany vedou do uměle přidaného kořene. Klauzule hornovské formule pak odpovídá jednotkovým tokům do kořene a nejvýše jednomu jednotkovému toku po hraně z kořene (pozitivní literál).

Pro skrytě hornovské formule se použije matice T, která má nenulové prvky pouze na diagonále a to ± 1 (1 je na pozicích těch proměnných, které je třeba komplementovat, abychom získali hornovskou formuli). □

Pro třídy extended horn a hidden extended horn není znám polynomiální algoritmus na rozpoznávání náležení formule do třídy (to souvisí s tím, že neznáme matici T^{-1}, resp. graf G_T). Pro testování splnitelnosti uvedeme algoritmus poté, co dokážeme základní vlastnosti těchto tříd. Budeme je totiž potřebovat pro odůvodnění korektnosti algoritmu.

Tvrzení 3.5.5. převážně z [8] *Třída hidden extended horn je uzavřená na odebrání klauzule, částečné dosazení, komplementaci proměnné. Není uzavřená na odebrání literálu, komplementaci literálu a spojení klauzulí.*

Důkaz. Třída je uzavřená na odebrání klauzule, protože její odebrání odpovídá odebrání řádku M_F — toku (tedy uvolní podmínky). Výsledná formule tak určitě bude mít vlastnosti rozšířené hvězdy se shodným grafem G_T.

Částečné dosazení odpovídá odebrání sloupců odpovídajících proměnných (proměnná se ve formuli již nevyskytuje) a řádků odpovídajících splněným klauzulím z matice M_F. Získanou matici označme M'_F (a odpovídající formuli F'). Odebrání řádku je popsáno při odůvodňování uzavřenosti na odebrání klauzule. Díky němu formule F' ze třídy hidden extended horn nevypadne. K odebírání sloupce musíme adekvátně upravit graf G_T (matici T^{-1}) — zde se užije kontrakce hrany (při odebrání k-tého sloupce M_F kontrahujeme tu hranu, které odpovídá k-tý řádek T^{-1}). Graf G'_T vzniklý po těchto úpravách je opět strom a dokazuje, že $M_{F'}$ má opět vlastnosti rozšířené hvězdy.

K důkazu uzavřenosti na komplementaci k-té proměnné (k-tý sloupec M_F) vytvoříme z G_T novou hvězdu tím, že opačně zorientujeme hranu odpovídající proměnné x (k-tý řádek T^{-1} vynásobíme -1).

Důvodem, proč je třída hidden extended horn neuzavřená na odebrání literálu je to, že změna nenuly na nulu v matici M_F může přerušit tok po nějaké cestě a tak nechat vzniknout vrcholu s přebytkem. Formule $F = (x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ je hidden extended horn. Její rozšířenou hvězdou je orientovaná cesta se třemi hranami (a žádnou jinou rozšířenou hvězdu nemá). Když F zopakujeme čtyřikrát po sobě, získáme opět hidden horn formuli (se stejnou hvězdou). Odebíráním literálu ale můžeme dostat formuli $F' = (x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)\&(x_1 \vee x_2)\&(\neg x_1 \vee \neg x_2)\&(x_2 \vee x_3)\&(\neg x_2 \vee \neg x_3)\&(x_1 \vee x_3)\&(\neg x_1 \vee \neg x_3)$, která není hidden extended horn. Nezávisle na přiřazení proměnných hranám cesty a pozici kořene totiž bude jedna z kvadratických klauzulí odpovídat toku s oběma dílčími toky vedoucími do vrcholu odlišného od kořene. Tyto vrcholy tudíž budou mít kladné přebytky, což poruší podmínky pro to, že je příslušný graf rozšířenou hvězdou pro tuto formuli.

Pro důkaz neuzavřenosti na komplemetaci literálu a spojení klauzulí nejdříve ukážeme, že formule $G = (x_1 \vee x_2)\&(x_1 \vee \neg x_2)\&(\neg x_1 \vee x_2) \vee (\neg x_1 \vee \neg x_2)$ nepatří do třídy hidden extended horn. To znamená, že neexistuje žádný stom G_T se třemi vrcholy (matice T^{-1} je rozšířená o jeden vrchol). Takový strom nemůže být nic jiného než nějakým způsobem zorientovaná cesta. Rozebereme dva případy:

- Kořen je prostřední vrchol cesty: Řádky M_G představují všechny čtyři kombinace ± 1. Bez ohledu na orientaci hran G_T bude vždy jeden z toků (řádků M_G) odpovídat dvěma jednotkovým tokům z kořene. Oba konce cesty tak budou mít přebytek 1, což narušuje požadavky na toky v rozšířené hvězdě.

- Kořen je jedním z konců cesty: Alespoň jeden řádek M_G (klauzule) odpovídá jednotkovému toku po obou hranách do středního vrcholu. V tomto vrcholu je přebytek 2, což vede k porušení požadavku na to, aby G_T byla rozšířená hvězda.

Nyní tedy ukážeme, že třída není uzavřená na komplementaci literálu. Formule $(\neg x_1 \vee \neg x_2)$ je hornovská, tedy je i hidden extended horn. Když ji dáme čtyřikrát za sebe do konjunkce, zůstane hidden extended horn. Komplementací literálu z takto vzniklé formule můžeme získat formuli G, o které jsme právě dokázali, že není hornovská.

Podobně lze dokázat i že třída hidden extended horn není uzavřená na spojení dvou formulí. Formule $G_1 = (x_1 \vee x_2)\&(x_1 \vee \neg x_2)$ a $G_2 = (\neg x_1 \vee x_2)\&(\neg x_1 \vee \neg x_2)$ jsou skrytě hornovské, a tedy i hidden exteded horn. Formule vzniklá jejich konjunkcí je G, která není hidden extended horn. □

Poznámka 3.5.6. Inkluze skrytě hornovské $\subsetneq$ hidden extended horn je ostrá. To plyne např. z toho, že třída hidden extended horn není uzavřená na odebrání literálu, kdežto třída skrytě hornovských formulí ano.

Abychom ospravedlnili korektnost algoritmu budeme, potřebovat uzavřenost třídy hidden extended horn na částečné dosazení. Tu užijeme k argumentaci, že po částečném dosazení provedeném v unitprop zůstane formule ve třídě hidden extended horn. Protože unitprop provádí pouze vynucená dosazení, je F' splnitelná, právě když je splnitelná F.

Algoritmus 6 Algoritmus pro hidden extended horn formule [13]

Vstup: Hornovská formule F v CNF.
Výstup: „splnitelná“ nebo „nesplnitená“.

```
1: (F', t) := unitprop(F)
2: if F obsahuje prázdnou klauzuli then
3:    return „nesplnitelná“
4: else
5:    return „splnitelná“
6: end if
```

Pokud F' obsahuje prázdnou klauzuli, je nesplnitelná. Zbývá tedy odůvodnit, proč je hidden extended horn formule F' se všemi klauzulemi alespoň kvadratickými vždy splnitelná. To proto, že systém nerovností pro F' je splněn vektorem $\vec{x'} = (\frac{1}{2}, \ldots, \frac{1}{2})$ — viz následující odstavec. Nyní použijeme na tento vektor $\vec{x'}$, systém nerovností a matici T'^{-1} (jejíž existenci zaručuje náležení F' do třídy hidden extended horn) větu 3.5.1 a dostáváme, že existuje celočíselné řešení této úlohy lineárního programování (a tedy splňující ohodnocení).

Jediné, co ještě nemusí být zřejmé, je proč vektor $\vec{x'}$ řeší daný systém nerovností. S nerovnostmi $\vec{x'} \geq \vec{0}$ a $-\vec{x'} \geq -\vec{e}$ jistě problém není. Podívejme se tedy na nerovnosti $M_{F'} \cdot \vec{x'} \geq \vec{e} - n(\vec{M_{F'}})$. Označme si $\vec{m}^T$ vybraný řádek matice $M_{F'}$, p počet hodnot 1 v tomto řádku a n počet hodnot -1 v tomto řádku (tj. stejné číslo jako příslušná složka $n(\vec{M_{F'}})$). Požadovaná nerovnost pak lze psát jako:

$$\vec{m^T} \cdot \vec{x'} = \frac{1}{2}(p - n) \geq 1 - n$$

a upravit na nerovnost

$$p + n \geq 2$$

a ta je splněna, protože jsme předpokládali, že v každé klauzuli jsou alespoň dva literály (na řádku $M_{F'}$ jsou alespoň dvě hodnoty ± 1).

Čtenáře by ještě mohlo zajímat, jak získat konkrétní splňující ohodnocení. To lze udělat pomocí známé techniky z úvah o využití černé skříňky řešící SAT (pro nás bude černou skříňku zastupovat uvedený algoritmus). K formuli F přidáme klauzuli x_1, ta vynucuje ohodnocení $v(x) = 1$. Pokud je formule $F\&x_1$ nesplnitelná, zkusíme stejný postup s $\neg x_1$ (a ohodnocení $v(x_1) = 0$). Je-li $F\&x_1$ splnitelná, znamená to, že existuje splňující ohodnocení přiřazující x_1 hodnotu 1. Toto ohodnocení si zapamatujeme, položíme $F := F\&x_1$ a postup opakujeme pro další proměnnou. Pokud jsou obě možnosti nesplnitelné, znamená to, že je formule nesplnitelná.

Postup lze zefektivnit pokud do dalších iterací používáme formule již po provedení jednotkové rezoluce. To vede k algoritmu SLUR uvedenému v následující části (ten lze tedy také užít k testování splnitelnosti hidden horn formulí).

Tvrzení 3.5.7. *Třída hidden extended horn je v inkluzi neporovnatelná se třídami kvadratických a matched formulí.*

Důkaz. Kvadratické $\not\subseteq$ Hidden extended horn: např. formule $F = (x_1 \vee x_2)\&(x_1 \vee \neg x_2)\&(\neg x_1 \vee x_2) \vee (\neg x_1 \vee \neg x_2)$ z tvrzení 3.5.5.

Hidden extended horn $\not\subseteq$ Kvadratické: stačí vzít libovolnou hornovskou formuli, která není kvadratická.

Matched $\not\subseteq$ Hidden extended horn: Formule $(x_1 \vee \neg x_2 \vee x_3)\&(x_1 \vee \neg x_2 \vee x_4)\&(x_1 \vee \neg x_2 \vee x_5)\&(\neg x_1 \vee \neg x_2 \vee x_6)$ je Matched (x_3, x_4, x_5 a x_6 zajistí existenci párování). Není ale hidden extended horn, protože třída hidden extended horn je uzavřená na částečné dosazení, ale dosazením hodnoty 0 za x_3, x_4, x_5 a x_6 vznikne výše uvedená formule F z tvrzení 3.5.5, o níž víme, že není hidden extended horn.

Hidden extended horn $\not\subseteq$ Matched: stači vzít libovolnou hornovskou formuli s více klauzulemi než proměnnými — např. $(\neg x_1 \vee \neg x_2)\&(x_1 \vee \neg x_2)\&(\neg x_1 \vee x_2)$. □

Pozorování 3.5.8. *Hidden extended horn není podtřídou třídy Balanced (tj. Hidden extended horn $\not\subseteq$ Balanced).*

Důkaz. V předcházející kapitole (tvrzení 3.4.2) jsme ukazovali hornovskou formuli, která není balanced. Tu můžeme použít i tady (hornovská je i hidden extended horn). □

Poznámka 3.5.9. Pro inkluzi Balanced $\not\subseteq$ Hidden extended horn se nepodařilo vymyslet protipříklad. Zůstává tak zatím otevřená.

Kapitola 4

Pokročilejší třídy s polynomiálně řešitelným SATem

4.1 SLUR

Třída SLUR (Single Lookahead Unit Resolution) byla zavedena v článku Schlipf a kol. [16]. Je definována pomocí nedeterministického algoritmu podobného Davis-Putnamově proceduře (viz [9]). V pozdějším článku Franca a Van Geldera [11] je k definici použita mírně upravená procedura (symetričtější vzhledem ke komplementaci proměnné), jíž budeme užívat i my. Třída je podle této definici sice menší v inkluzi, ale přirozenější. Konkrétně se jedná o to, že není preferováno ohodnocení jednou pravdivostní hodnotou před druhou (řádek 18 algoritmu).

Definice 4.1.1. [11] SLUR je třída těch formulí, na kterých algoritmus SLUR (uveden na straně 35) při žádné volbě pořadí proměnných a volbách dosazených hodnot nevrátí „selhalo“.

Je snadno vidět, že algoritmus testování splnitelnosti má pro formule třídy SLUR časovou složitost $O(n^2)$ (implementujeme-li funkci unitprop v lineárním čase). Nyní ukážeme některé vlastnosti této třídy.

Tvrzení 4.1.2. *Třída SLUR je uzavřená na komplementaci proměnné.*

Důkaz. Stačí, když ukážeme uzavřenost třídy na komplementaci jedné proměnné. Budeme postupovat sporem. Nechť F je formule třídy SLUR, x její proměnná a F' vznikne z F komplementací proměnné x, která již není ve třídě SLUR.

Vezměme výpočet (volbu pořadí proměnných a dosazených hodnot), pro který algoritmus SLUR na formuli F selhal. Vytvoříme-li z něj výpočet, který se bude lišit pouze tím, že za x dosadí opačnou hodnotu, získáme korektní výpočet pro formuli F. Existence tohoto výpočtu je ale ve sporu s tím, že F je ve třídě SLUR (při této volbě pořadí proměnných a dosazených hodnot selže). □

Tvrzení 4.1.3. *Třída SLUR není uzavřená na komplementaci literálu.*

Důkaz. Uvažme formuli $(x_1 \vee x_2)\&(x_1 \vee \neg x_2)\&(\neg x_1 \vee x_2)\&(a \vee \neg x_1 \vee x_2)$. Pokud komplementujeme poslední literál, pak je každý výpočet, který v prvním krouku přiřadí proměnné a hodnotu 0 selhávající, protože $(x_1 \vee x_2)\&(x_1 \vee \neg x_2)\&(\neg x_1 \vee x_2)\&(\neg x_1 \vee \neg x_2)$ není splnitelná. □

Algoritmus 7 SLUR

Vstup: Formule F v CNF na proměnných $x_1, x_2, \ldots, x_n$

Výstup: Splnující částečné pravdivostní ohodnocení proměnných z F, „nesplnitená“, nebo „selhalo“

```
1: F, t := unitprop(F)
2: if F obsahuje prázdnou klauzuli then
3:   return „nesplnitelná“
4: end if
5: while F není prázdná množina klauzulí do
6:   v := libovolná proměnná vyskytující se v F
7:   (F_1, t_1) := unitprop( F ∪ {{¬v}} )
8:   (F_2, t_2) := unitprop( F ∪ {{v}} )
9:   if ∅ ∈ F_1 and ∅ ∈ F_2 then
10:    return „selhalo“
11:  else if ∅ ∈ F_1 then
12:    F := F_2
13:    t := t ∪ t_2
14:  else if ∅ ∈ F_2 then
15:    F := F_1
16:    t := t ∪ t_1
17:  else
18:    Vyber libovolnou z následujících možností:
19:    1. F := F_1, t := t ∪ t_1
20:    2. F := F_2, t := t ∪ t_2
21:  end if
22: end while
23: return t
```

Tvrzení 4.1.4. *Třída SLUR je uzavřená na částečné dosazení.*

Důkaz. Tvrzení ukážeme pro dosazení hodnoty jedné proměnné (dále lze užít princip indukce).

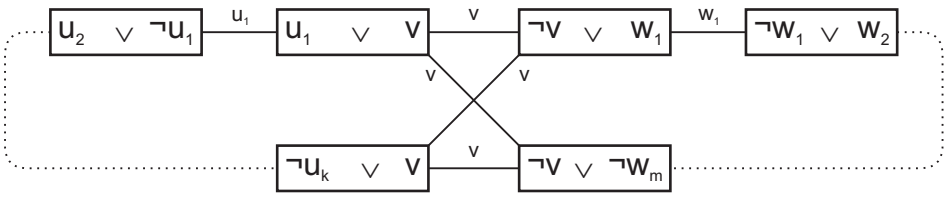
Vezměme libovolnou formuli F z třídy SLUR a proměnnou x, která se v ní vyskytuje (v případě, kdy se x v F nevyskytuje, platí tvrzení triviálně). Rozebereme dva případy podle toho, kdy byla ve výpočtu nad F (formulí před částečným dosazením) přiřazena proměnné x její hodnota:

a) algoritmus přiřadil proměnné x její hodnotu již před vstupem do while-cyklu (tedy v úvodním volání funkce unitprop): To znamená, že se buď proměnná vyskytovala v jednotkové klauzuli, nebo že její hodnota byla vynucena zprostředkovaně (posloupností rezolucí). V případě, že je hodnota určená částečným dosazením shodná, tvrzení zřejmě platí. V opačném případě, je funkcí unitprop odvozen spor a tvrzení také platí.

b) hodnota je přiřazena v průběhu výpočtu while-cyklu: Protože F je SLUR, musí algoritmus úspěšně skončit při jakémkoliv pořadí volby proměnných. Můžeme tedy zafixovat volbu proměnné x jako první vybrané. Díky tomu, že F je SLUR, při libovolném pořadí dalších proměnných algoritmus neselže. To navíc platí pro obě hodnoty přiřazení proměnné x (v každém z případů je buď okamžitě vyvozen spor pomocí funkce unitprop - a to nejvýše pro jednu z možných hodnot, nebo lze pokračovat ve výpočtu). Protože přiřazení hodnoty proměnné je prováděno pomocí konjunkce formule s jednotkovou klauzulí a volání unitprop, můžeme použít předchozí úvahy z bodu a). □

V článku [11] je uvedena struktura, jejíž výskyt v rezolučním grafu (definovaném na str. 13) za předpokladu, že všechny v, u_i, w_j jsou navzájem různé proměnné, a za předpokladu, že ve formuli nejsou další klauzule, jež by do grafu přidaly další cestu, zajišťuje, že formule nebude SLUR. Tuto strukturu uvádíme na obrázku 4.1 v mírně pozměněné podobě (původní struktura byla uzpůsobená počtem klauzulí pro snazší pravděpodobnostní odhady). Každý vrchol grafu představuje klauzuli, hrany právě jednu konfliktní proměnnou (a tedy možnost rezoluce jí spojených klauzulí). V klauzulích se mohou vyskytovat další literály, ale ty nesmí přidat další komplementární pár.

To, že formule s danou strukturou není SLUR, lze jednoduše nahlédnout z toho, že neobsahuje jednotkové klauzule (algoritmus na ní musí zvolit hodnotu pevně přiřazenou proměnné) a není splnitelná (každý výpočet odvodí prázdnou klauzuli). Vyberme jako první třeba proměnnou v. Dosadíme-li za ní hodnotu 0, levá polovina dává spor. Dosadíme-li 1 dává spor pravá polovina.



Obrázek 4.1: Criss-cross smyčka

Nyní použijeme uvedenou strukturu k důkazu několika dalších tvrzení o třídě SLUR.

Tvrzení 4.1.5. *Jsou-li F_1 a F_2 SLUR formule, formule $F_1\&F_2$ nemusí být SLUR, tj. SLUR není uzavřena na spojení dvou formulí.*

Důkaz. Myšlenkou tohoto důkazu je spojit dvě formule (odpovídající dvěma cyklům), aby v rezolučním grafu vznikla uvedená struktura (dvě hamiltonovské kružnice).

Příkladem takových formulí může být $F_1 = (v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)$ a $F_2 = (\neg v \vee w_1)\&(\neg w_1 \vee w_2)\&(\neg w_2 \vee \neg v)$. Obě tyto formule jsou SLUR (ukážeme pro F_1, pro F_2 to pak plyne z uzavřenosti na komplementaci proměnné):

1. Nechť je jako první vybrána proměnná v. Pokud se za ni snažíme dosadit hodnotu 0, jsou přímo vynuceny hodnoty proměnných u_1 a u_2 po řadě 1 a 0, které dávají spor u prostřední klauzule. Je tak u proměnné v vynucena hodnota 1 a do dalšího kroku zbyde pouze prostřední klauzule, která už je jistě SLUR.

2. Nechť je nyní jako první vybrána proměnná u_1 (případ u_2 je symetrický). Pokud jí přiřadíme hodnotu 0, vynutí se pro v hodnota 1 a tím se splní celá formule. Jestliže jí přiřadíme hodnotu 1, vynutí se hodnota 1 i pro proměnnou u_2 a ta vynutí hodnotu 1 proměnné v, čímž se formule splní.

Spojením $F_1\&F_2$ dostaneme strukturu odpovídající criss-cross smyčce uvedené na obrázku 4.1. Spojení tedy nebude SLUR.

Ještě poznamenejme, že budou-li dvě formule SLUR na disjunktních množinách proměnných, je SLUR i jejich konjunkce (ale toto není nutná podmínka). □

Příklad 4.1.6. Vlastnost „býti SLUR“ je vlastností konkrétní formule, nikoliv funkce, kterou formule reprezentuje, jak ukazují následující formule: $F_1 = x_1\&\neg x_1$ a $F_2 = (x_1 \vee x_2)\&(\neg x_1 \vee x_2)\&(x_1 \vee x_2)\&(\neg x_1 \vee \neg x_2)$. Obě jsou nesplnitelné (vyjadřují nulovou funkci) a přitom F_1 je SLUR a F_2 není (výpočet selže při obou pořadích proměnných).

Platí však následující tvrzení.

Tvrzení 4.1.7. [16] *Je-li formule F SLUR a F' je její logický důsledek, pak je SLUR i formule $F\&F'$.*

Důkaz. Tvrzení je zřejmé. Přidané klauzule případně pouze urychlují výpočet (dříve vynutí správnou hodnotu proměnné). □

Tvrzení 4.1.8. *Třída SLUR není uzavřená na odebrání klauzule z formule.*

Důkaz. Formule $F = s\&(s \vee x_1 \vee x_2)\&(s \vee \neg x_1 \vee x_2)\&(s \vee x_1 \vee x_2)\&(s \vee \neg x_1 \vee \neg x_2)$ je SLUR (jednotková rezoluce na řádce 1 algoritmu SLUR dosadí za s hodnotu 1, čímž splní všechny klauzule).

Odstraníme-li jednotkovou klauzuli s, vznikne formule, která SLUR není (pokud pro vzniklou formuli vybereme jako první proměnnou s a přiřadíme jí hodnotu 0, získáme formuli F_2 z předchozího příkladu).

Jiným případem takové formule je $F = F'\&x\&\neg x$, kde F' je formule jež není SLUR a neobsahuje proměnnou x (např. F_2 z výše uvedeného příkladu). Tato je SLUR (z posledních dvou klauzulí je odvozen spor a formule je označena za nesplnitelnou), odebereme-li poslední klauzuli, získáme formuli, na které algoritmus selže. □

Tvrzení 4.1.9. *Třída SLUR není uzavřená na odebrání literálu z klauzule.*

Důkaz. Příkladem buď formule $F = (v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg v \vee w_1 \vee a)\&(\neg w_1 \vee w_2)\&(\neg w_2 \vee \neg v \vee \neg a)$, která je SLUR:

- Nejdříve nechť je jako první vybrána proměnná v. Zkusíme-li za ní dosadit 0, tím je vynuceno dosazení 1 za u_1 a 0 za u_2, to dá nad druhou klauzulí spor. Dosadíme tedy 1, tím pádem nám zbyde formule $(\neg u_1 \vee u_2)\&(w_1 \vee a)\&(\neg w_1 \vee w_2)\&(\neg w_2 \vee \neg a)$. O první klauzuli se nemusíme starat, protože je na jiných proměnných než zbytek — za jednu proměnnou dosadíme a pro druhou je hodnota vynucena (pokud klauzule nebyla splněna již dosazením za první proměnnou). Na zbylé tři klauzule se můžeme dívat jako na cyklus implikací (tak jako v části o kvadratických formulích). Pokud dosadím jednu hodnotu, budou zbylé vynuceny.

- Nechť je jako první vybrána proměnná u_1. Dosadíme-li za u_1 nulu, je vynuceno dosazení 1 za v a to vede na předchozí řetězec úvah. Dosadíme-li za u_1 jedničku, je vynuceno ohodnocení u_2 jedničkou a v jedničkou, což zase vede na předchozí bod. Situace pro volbu u_2 jako první proměnné k dosazování je obdobná.

- Nechť je nyní jako první proměnná zvolena a. Dosazením hodnoty 0 dostáváme $(v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg v \vee w_1)\&(\neg w_1 \vee w_2)$. Ať dosadíme za jakoukoliv další proměnnou, vždy buď dospějeme k vynucení hodnoty 1 za v, nebo ke sporu. Pokud bychom za a dosazovali 1, dostaneme $(v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg w_1 \vee w_2)\&(\neg w_2 \vee \neg v)$, kde je situace obdobná.

- Poslední možností je vybrat jako první proměnnou w_1 (případ w_2 je obdobný). Dosazená hodnota 0 dává $(v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg v \vee a)\&(\neg w_2 \vee \neg v \vee \neg a)$. Ohodnocení v, u_1 nebo u_2 jako další proměnné vede k úvahám prvních dvou bodů. Ohocnocení w_2 jako druhé proměnné buď odstraní, nebo zkrátí poslední klauzuli (a vede na známé úvahy). Ohodnocení a jako druhé proměnné vynucuje ohodnocení hodnotou 1 (jinak je vynucené dosazení 0 za v a to dává spor zjistitelný v unitprop — nepoužije se) a to vede k formuli $(v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg w_2 \vee \neg v)$, na kterou můžeme použít dřívější úvahy.

  Pokud bychom za a dosadili 0, získáme $(v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg v \vee w_1)\&(\neg w_1 \vee w_2)$, kde opět můžeme použít téže úvahy.

Odebereme-li z ní literál a (případně $\neg a$), vznikne formule uvedená v tvrzení 4.1.5 o neuzavřenosti na konjunkci dvou SLUR formulí (navíc v ní zůstane pouze jeden literál $\neg a$ resp. a, který neovlivní její vlastnosti — není SLUR): máme totiž formuli $(v \vee u_1)\&(\neg u_1 \vee u_2)\&(\neg u_2 \vee v)\&(\neg v \vee w_1)\&(\neg w_1 \vee w_2)\&(\neg w_2 \vee \neg v \vee \neg a)$, po dosazení hodnoty 1 za proměnnou a jednotková rezoluce nic neodvodí a jsme u formule, o které se hovoří v citovaném tvrzení. □

Lemma 4.1.10. *Rozhodování $F \in$ SLUR patří do třídy co-NP.*

Důkaz. Použijeme důkaz náležení do třídy co-NP pomocí certifikátu. Tím bude pořadí proměnných, na kterých výpočet selhává, a seznam vybraných možností (dojde-li k vykonání řádku 18 algoritmu SLUR).

Tento certifikát je zjevně polynomiální vzhledem k velikosti formule. Jeho ověření lze provést v polynomiálním čase simulací výpočtu algoritmu SLUR s příslušnými volbami a pořadím proměnných. □

Nyní se pokusíme třídu SLUR zvětšit pomocí jednoduché myšlenky nehledat spor pro volbu pouze po bezprostředním dosazení jedné hodnoty, ale dosazovat za více hodnot. Výpočet se tak sice zpomalí, ale dostamene mírně větší třídu, jak ukážeme v následujícím.

Definice 4.1.11. SLUR(i) je třída logických formulí, na kterých algoritmus SLUR(i) ze strany 40 nevrátí „selhalo“ při žádné z nedeterministických voleb množiny A (na řádku 10) a voleb částečných dosazení (na řádku 26).

Počet všech kombinací přímosti/negovanosti i-tice literálů je roven 2^i. For–cyklus na řádku 12 tak spotřebuje čas $O(n \cdot 2^i)$ (za předpokladu, že je funkce unitprop implementována v lineárním čase). Zbytek těla while-cyklu se do tohoto odhadu také jistě vejde. Počet běhů while-cyklu je shora omezen počtem proměnných n. Celý algoritmus SLUR(i) tudíž pracuje v čase $O(n^2 \cdot 2^i)$.

Třída SLUR(1) odpovídá původní definici třídy SLUR s tím, že v případě, kdy v prvním běhu while cyklu selžou obě větve, vrátíme „nesplnitelná“. Nyní ukážeme vztahy třídy SLUR(i) s ostatními třídami formulí.

Tvrzení 4.1.12. *Pro každé i existuje formule F, že $F \in SLUR(i+1) \setminus SLUR(i)$.*

Důkaz. Takovou formulí je například identická nula vyjádřená jako CNF na $i+2$ proměnných $V = \{x_1, \ldots, x_{i+2}\}$ obsahující všechny kombinace přímých a negovaných literálů:

$$\bigwedge_{P \subseteq V} \left(\bigvee_{v \in P} v \vee \bigvee_{v \in V \setminus P} \neg v \right).$$

Když do této formule dosadíme za libovolnou i-tici proměnných, některé klauzule vymizí a zůstane nám sporná formule dvou proměnných (např. dosadíme-li za $x_1, \ldots, x_i$, zbude nám $(x_{i+1} \vee x_{i+2}) \& (x_{i+1} \vee \neg x_{i+2}) \& (\neg x_{i+1} \vee x_{i+2}) \& (\neg x_{i+1} \vee \neg x_{i+2})$), na které unitprop neodvodí spor. Pokud dosadíme za libovolnou $i+1$-tici, tak nám zbyde sporná formule typu $x \& \neg x$, na které unitprop spor odvodí. □

Tvrzení 4.1.13. *Pro každé i je třída Matched formulí neporovnatelná s třídou SLUR(i).*

Důkaz. SLUR(i) $\not\subseteq$ Matched: plyne z toho, že ve třídě SLUR(i) existují formule, které jsou nesplnitelné (ty jistě nepatří do třídy Matched).

Matched $\not\subseteq$ SLUR(i): vezmeme formuli z důkazu tvrzení 4.1.12 tak, aby nebyla SLUR(i) a vytvoříme z ní Matched formuli tak, že do každé klauzule přidáme novou proměnnou $(y_1, \ldots)$ v přímé formě. Tyto proměnné budou tvořit totální párování (a formule tedy bude Matched). Pokud za každou z proměnných y_i (kterých bude 2^i, tj. více než i) dosadíme 0, zbyde nám formule z uvedeného tvrzení, která není splnitelná a tedy na ní algoritmus SLUR(i) selže. □

Tvrzení 4.1.14. *Pro každé i je třída SLUR(i) neporovnatelná se třídou kvadratických formulí.*

Důkaz. Položme $F = (x_1 \vee x_2) \& (\neg x_1 \vee x_2) \& (x_1 \vee \neg x_2) \& (\neg x_1 \vee \neg x_2)$ a $G_j = (y_{j,1} \vee y_{j,2})$.

Pak formule $G_1 \& F$ není SLUR(1): stačí zvolit za proměnnou $y_{1,1}$, zbyde formule F a ta je nesplnitelná. Algoritmus tedy vrátí „selhalo“.

Obdobnou formuli lze definovat pro každé i: $G_1 \& \ldots \& G_i \& F$ a v prvním kroku vybrat proměnné $y_{1,1}, y_{2,1}, \ldots, y_{i,1}$.

Na druhou stranu existují SLUR formule, které nejsou kvadratické. □

Algoritmus 8 SLUR(i)

Vstup: Formule F v CNF na proměnných $x_1, x_2, \ldots, x_n$

Výstup: Splňující částečné pravdivostní ohodnocení proměnných z F, „nesplnitená“, nebo „selhalo“

```
1: F, t := unitprop(F)
2: if F obsahuje prázdnou klauzuli then
3:   return „nesplnitelná“
4: end if
5: j := 0
6: while F není prázdná množina klauzulí do
7:   j := j + 1
8:   Q := prázdná fronta
9:   l_Q := 0
10:  A := množina i proměnných obsažených ve formuli F (pokud F obsahuje
     méně proměnných, pak množina všech proměnných z F)
11:  B := {{{l_1}, ..., {l_|A|}} : l_1, ..., l_|A| jsou literály nad všemi proměnnými z A}
     {Každý prvek B je formule složená z jednotkových klauzulí nad proměnnými
     různými proměnnými z A (proto tolik možinových závorek)}
12:  for all S ∈ B do
13:    F_S, t_S := unitprop(F ∪ S)
14:    if F_S neobsahuje prázdnou klauzuli then
15:      přidej F_S, t_S do Q
16:      l_Q := l_Q + 1
17:    end if
18:  end for
19:  if l_Q = 0 then
20:    if j = 1 then
21:      return „nesplnitelná“
22:    else
23:      return „selhalo“
24:    end if
25:  else
26:    F, t_vybrán := náhodně vybraný prvek z Q
27:    t := t ∪ t_vybrán
28:  end if
29: end while
30: return t
```

Poznámka 4.1.15. O kvadratické formule lze libovolnou třídu SLUR(i) i původní SLUR rozšířit jednoduchým způsobem. Stačí si zapamatovat, že formule byla na počátku kvadratická a v případě, že algoritmus selže, je formule nesplnitelná (což plyne z toho, že kvadratické formule lze řešit pomocí jednotkove rezoluce – viz sekce o kvadratických formulích).

Tvrzení 4.1.16. *Třída SLUR obsahuje třídu hidden extended horn.*

Důkaz. Plyne z toho, že algoritmus SLUR vyřeší splnitelnost pro každou hidden extended horn formuli — viz diskuze pod algoritmem 6 na straně 32. □

Tvrzení 4.1.17. *Třída SLUR obsahuje třídu Balanced.*

Důkaz. Porovnáním algoritmu 4 (strana 21) a algoritmu SLUR. □

4.2 q-Horn

Definice 4.2.1. Nechť V je množina proměnných. *Zobecněné pravdivostní ohodnocení proměnných* je zobrazení $u : V \to [0, 1]$. Toto ohodnocení lze rozšířit na literály požadavkem $\bar{u}(x) + \bar{u}(\neg x) = 1$ a klauzule $\bar{u}(C) = \sum_{l \in C} \bar{u}(l)$.

Ohodnocení $\bar{u}$ nazveme *přípustným pro klauzuli* C, jestliže $\bar{u}(C) \leq 1$.

Formule F patří do třídy *q-Horn*, jestliže existuje zobecněné ohodnocení u takové, že $\bar{u}$ je přípustné pro všechny klauzule formule F.

Třídu q-Horn lze ekvivalentně definovat pomocí tzv. complexity indexu ([5]). S ohledem na následující kapitolu definujeme complexity index pro ohodnocení (vektory $\vec{x}$) s prvky z $[-1, 1]$.

Definice 4.2.2. *Complexity index* pro formuli F s incidenční maticí M_F (viz definice 2.3.4) je řešení následujícího lineárního programu:

$$\begin{cases} \min Z \\ M_F \cdot \vec{x} \geq L - 2Z\vec{e} \\ \vec{x} \in [-1, 1]^n \end{cases}$$

kde L je vektor mající v i-té složce délku i-té klauzule, $\vec{e}$ je vektor samých jedniček a $Z \in \mathbb{R}$.

Definice 4.2.3. Formule F je ve třídě q-Horn, jestliže má complexity index $Z \leq 1$.

Abychom nahlédli, že obě definice vedou ke stejné třídě formulí, zavedeme si ohodnocení $t : V \to [-1, 1]$. Pro převod z jednoho ohodnocení do druhého použijeme vztah $u(x) = \frac{1}{2}(t(x)+1)$. Požadavku $\bar{u}(x)+\bar{u}(\neg x) = 1$ pak bude odpovídat požadavek $\bar{t}(x) + \bar{t}(\neg x) = 0$. Dosazením do požadavku přípustnosti pro klauzule

$$\bar{u}(C) = \sum_{l \in C} l \leq 1$$

dostáváme požadavek

$$\frac{1}{2} \sum_{l \in C} t(l) + \frac{L_C}{2} \leq 1$$

kde L_C je délka klauzule C. Označíme-li $\bar{t}(C) := \sum_{l \in C} t(l)$ a upravíme rovnici, dostaneme vztah $\bar{t}(C) \leq 2 - L_C$, a po vynásobení hodnotou -1 dostáváme $-\bar{t}(C) \geq L_C - 2$. Nyní si uvědomíme, že lineární program definující complexity index bere do úvahy všechny vektory $\vec{x} \in [-1, 1]^n$ (tj. zkouší všechna ohodnocení t) a můžeme tedy pro účely důkazu ekvivalence definic umazat mínus na levé straně. Dále si uvědomíme, že součin řádku M_F s $\vec{x}$ odpovídá aplikaci ohodnocení na klauzuli, tj. $\bar{t}(C_i)$ (kde C_i je klauzule odpovídající i-tému řádku). Nyní už je ekvivalence obou definic zřejmá.

Dále je možno definovat třídu q-Horn pomocí tzv. monotónních matic (ekvivalence s předchozími definicemi je dokázána v [5]).

Definice 4.2.4. Nechť F je formule a M_F její incidenční matice. *Monotónní dekompozicí matice* M_F rozumíme matici M'_F získanou z M_F permutací řádků a sloupců, případně násobením sloupců hodnotou -1 (komplementace proměnné), následujícího tvaru:

$$\left(\begin{array}{c|c} A_1 & 0 \\ \hline D & A_2 \end{array}\right)$$

kde A_1 má nejvýše jednu hodnotu 1 v každém řádku, D obsahuje pouze hodnoty -1 a 0 a A_2 je libovolná matice s prvky $\{0, \pm 1\}$.

Maximální monotónní dekompozice je taková, ve které je velikost A_1 maximalizována.

Jestliže maximální monotónní dekompozice M_F neobsahuje v části A_2 více než dvě nenuly na řádek, pak je F ve třídě q-Horn.

Na základě této dekompozice lze vytvořit lineární algoritmus pro test náležení formule do třídy q-Horn. Algoritmem z knihy [18] (resp. zprávy [19]) v lineárním čase sestrojíme maximální monotónní dekompozici a ověříme podmínku uvedenou v definici 4.2.4.

Jiný lineární algoritmus rozpoznávání formulí třídy je možné najít v článku [4].

V okamžiku, kdy máme F dekomponovánu, můžeme otestovat splnitelnost následujícím způsobem v lineárním čase. Algoritmem pro hornovské formule najdeme splňující ohodnocení hornovské části (to jsou řádky M_F odpovídající matici A_1). Je výhodné hledat splňující ohodnocení s maximálním možným počtem nul (pokud je sloupec ohodnocen na 0, jsou splněny i všechny klauzule, které mají v příslušném sloupci matice D nenulu – tj. -1). Pokud je tato hornovská část (podformule) nesplnitelná, je nesplnitelná celá formule (a končíme). Nyní odebereme všechny řádky odpovídající splněným klauzulím a sloupce incidentní s maticí A_1 (byly již ohodnoceny). Protože nám zbyly pouze řádky matice A_2 (příp. jen některé) a A_2 neobsahovala více než dvě nenuly na řádek, můžeme její splnitelnost otestovat algoritmem pro kvadratické formule. Případné splňující ohodnocení pak bude odpovídat sjednocení ohodnocení z hornovské části a kvadratické části algoritmu.

Tvrzení 4.2.5. převážně z [8] *Třída q-horn formulí je uzavřena na částečné dosazení, odebrání literálu i klauzule, komplementaci proměnné. Není uzavřená na spojení formulí a komplementaci literálu.*

Důkaz. Uzavřenost na částečné dosazení plyne z definice. Nejsnadněji je to vidět na definici pomocí complexity indexu (pro původní formuli je $\vec{x} \in [-1, 1]^n$, po částečném dosazení se některé proměnné zafixují).

Uzavřenost na odebrání literálu a klauzule je zase nejlépe vidět z první definice třídy. Odebrání klauzule C vede k vynechání celé podmínky $\sum_{\bar{u}(l)\in C} l \leq 1$. Odebrání literálu se projeví odstraněním jednoho (kladného) sčítance.

Pro komplementaci proměnné dokážeme uzavřenost následovně. Nechť původní formule F má přípustné ohodnocení u. A nechť x je komplementovaná proměnná a její komplementací vznikne formule F'. Pak pro F' je přípustné ohodnocení u' vzniklé z u následovně: $u'(y) = u(y)$ pro $y \neq x$ a $u'(x) = 1 - u(x)$.

Formule $F_1 = (x_1 \vee x_2 \vee x_3)$ a $F_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ jsou q-Horn. Pro první to dokazuje přípustné ohodnocení $u(x_i) = 0, i \in \{1, 2, 3\}$ a pro druhou můžeme argumentovat tak, že vznikne komplementací všech proměnných F_1. Formule vzniklá spojením obou $F_1 \& F_2$ ale už není q-Horn. K tomu by muselo existovat zobrazení $u : \{x_1, x_2, x_3\} \to [0, 1]$, pro které je zároveň $u(x_1) + u(x_2) + u(x_3) \leq 1$ a $\bar{u}(\neg x_1) + \bar{u}(\neg x_2) + \bar{u}(\neg x_3) \leq 1$ za podmínky $u(x) + \bar{u}(\neg x) = 1$. Sečtením uvedených nerovností a použitím podmínky dostáváme $1 + 1 + 1 \leq 1 + 1$, což není možné.

Pro neuzavřenost na komplementaci literálu použijeme formuli $(\neg x_1 \vee \neg x_2 \vee x_3) \& (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Ta je v q-Horn, jak dokazuje přípustné ohodnocení $u(x_1) = u(x_2) = 1$, $u(x_3) = \frac{1}{2}$. Komplementací jejích prvních dvou literálů získáme formuli z předchozího odstavce, která není v q-Horn. □

Pozorování 4.2.6. [8] *Třída q-Horn je vlastní nadtřídou třídy kvadratických formulí.*

Důkaz. Pro každou kvadratickou formuli lze užít ohodnocení všech proměnných hodnotou $\frac{1}{2}$. Pro každou klauzuli C pak jistě je $\bar{u}(C) \leq 1$, protože obsahuje nejvýše dva literály. Na druhou stranu jsme již viděli formule z q-Horn, které nebyly kvadratické. □

Pozorování 4.2.7. [8] *Třída q-Horn je vlastní nadtřídou třídy skrytě hornovských formulí.*

Důkaz. Nechť F je skrytě hornovská formule. Nechť dále S je množina proměnných, jejichž komplementací získáme z F hornovskou formuli. Pak ohodnocení $u(x) = 0$ pro $x \in S$ a $u(x) = 1$ pro $x \notin S$ je přípustné pro F a ta je tudíž hornovská. Ostrost inkluze plyne z toho, že třída skrytě hornovských formulí neobsahuje třídu kvadratických formulí, zatímco třída q-Horn ano. □

Tvrzení 4.2.8. *Třída q-Horn je v inkluzi neporovnatelná se třídou Balanced.*

Důkaz. q-Horn $\not\subseteq$ Balanced: Třída q-Horn obsahuje třídu hornovských formulí a o té víme, že je s Balanced neporovnatelná – příslušná formule, která je hornovská (a tedy i q-Horn) a není balanced je v tvrzení 3.4.2 na straně 24.

Balanced $\not\subseteq$ q-Horn: $(x_1 \vee x_2 \vee x_3) \& (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ je balanced (jak bylo ukázáno v 3.4.2), ale (jak už víme z důkazu tvrzení 4.2.5) není q-Horn. □

Tvrzení 4.2.9. *Třída q-Horn je v inkluzi neporovnatelná se třídou Matched.*

Důkaz. q-Horn $\not\subseteq$ Matched: Formule $x \& \neg x$ je q-Horn, ale není Matched.

Matched $\not\subseteq$ q-Horn: Opět lze užít formuli $(x_1 \vee x_2 \vee x_3) \& (\neg x_1 \vee \neg x_2 \vee \neg x_3)$: je Matched (vezmeme párování vybírající z každé klauzule jinou proměnnou), ale není q-Horn. □

Tvrzení 4.2.10. [8] *Třída q-Horn je v inkluzi neporovnatelná se třídou hidden extended horn.*

Důkaz. q-Horn $\not\subseteq$ hidden extended horn: Formule $(x_1 \vee \neg x_2 \vee x_3)\&(x_1 \vee \neg x_2 \vee x_4)\&(x_1 \vee \neg x_2 \vee x_5)\&(\neg x_1 \vee \neg x_2 \vee x_6)$ je q-Horn (stačí zvolit $u(x_1) = u(x_2) = u(x_3) = \frac{1}{2}$ a pro zbývající proměnné x položit $u(x) = 0$). Není však hidden extended horn (viz tvrzení 3.5.7 na straně 32, vztah Matched $\not\subseteq$ hidden extended horn).

Hidden extended horn $\not\subseteq$ q-Horn: Opět lze užít formule $(x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)$, o které už dávno víme, že není q-Horn. To, že je hidden extended horn jsme odůvodnili v důkazu tvrzení 3.5.5 (str. 30) v části o neuzavřenosti hidden extended horn na odebrání literálu. □

Tvrzení 4.2.11. *Pro každé i je třída q-Horn neporovnatelná se SLUR(i).*

Důkaz. $SLUR(i)$ $\not\subseteq$q-Horn: $(x_1 \vee x_2 \vee x_3)\&(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ je $SLUR$ (dosazení za libovolnou proměnnou jednu klauzuli odebere a druhou zkrátí na kvadratickou; formule o jedné klauzuli je pak vždy SLUR, každým dosazením je buď splněná nebo zkrácená a v případě, že je zkrácena na jednotkovou klauzuli, vynutí hodnotu, která jí splní). Tato formule ale není q-Horn (viz důkaz neuzavřenosti na spojení dvou formulí v tvrzení 4.2.5).

q-Horn $\not\subseteq$ $SLUR(i)$: jako příklad formule, která není SLUR(i), ale je q-Horn můžeme použít formuli z tvrzení 4.1.14 (a znalost, že každá kvadratická klauzule je q-Horn). □

4.3 Linear autarkies

Myšlenky vedoucí ke třídě LinAut se objevují v Kullmannově článku [14] (definice, ke které dospějeme, odpovídá článku [20] – podrobněji viz poznámka 4.3.8). Začneme definicí a základními výsledky z původního článku [14] (v poněkud odlišné symbolice).

Definice 4.3.1. Autarky je částečné ohodnocení v' ohodnocující alespoň jednu proměnnou, které splní každou klauzuli, v níž ohodnotí alespoň jednu proměnnou.

Není težké nahlédnout, že hledaní autarkies je NP-úplné: autarky lze totiž skládat a ohodnocení splňující celou formuli je jistě autarky – získali bychom tak algoritmus pro řešení SATu (dokud je formule neprázdná, najdi autarky a aplikuj na formuli, pokud autarky neexistuje, je formule nesplnitelná).

Třída LinAut je založena na myšlence, že jistý druh těchto částečných ohodnocení lze najít v polynomiálním čase.

Definice 4.3.2. Nechť $V \supseteq V' \neq \varnothing$. Částečné ohodnocení $v' : V' \to \{0, 1\}$ je linear autarky pro formuli F (v CNF), jestliže existuje váhová funkce $w : V \to \mathbb{Q}^+$ přiřazující každé proměnné kladné racionální číslo tak, že pro každou klauzuli $C \in F$ platí:

$$\sum_{l \in C, v'(l)=1} w(var(l)) \geq \sum_{l \in C, v'(l)=0} w(var(l))$$

kde $var(l)$ značí proměnnou v literálu l.

Poznámka 4.3.3. Pokud bychom se měli držet původní terminologie z článku [14], nazvali bychom výše uvedené jako simple linear autarky. Linear autarky by pak byla libovolná autarky získaná složením několika simple linear autarkies (která ovšem už nemusí být simple linear autarky).

Pozorování 4.3.4. [14] *Částečné ohodnocení v' splňující uvedenou nerovnost (linear autarky) je autarky.*

Důkaz. Vezměme klauzuli C obsahující literál l_1 takový, že $v'(l_1) = 0$ (klauzule, do které v' nic nedosazuje, nás nezajímají a případ, kdy je literál ohodnocen na 1 nemůže narušit to, že ohodnocení je autarky). Pak platí $\sum_{l \in C, v'(l)=1} w(var(l)) \geq \sum_{l \in C, v'(l)=0} w(var(l)) \geq w(var(l_1)) > 0$, a tedy v C musí existovat l_2, který je ohodnocen na 1 (přispívá do první sumy). □

V uvedeném článku je možno nalézt mnoho vlastností autarkies, které zde nebudeme potřebovat.

Nyní popíšeme, jak hledat lineární autarkies pomocí lineárního programování. K tomu budeme potřebovat následující tvrzení.

Tvrzení 4.3.5. [14] *Všechny linear autarkies pro F jsou přesně prvky množiny $L_F = \{\vec{x} \in \mathbb{Q}^n : M_F \cdot \vec{x} \geq \vec{0}, \vec{x} \neq \vec{0}\}$, interpretované následujícím způsobem:*

$$\begin{cases} \textit{i-tá proměnná je ohodnocena 1, jestliže } (\vec{x})_i > 0 \\ \textit{i-tá proměnná je ohodnocena 0, jestliže } (\vec{x})_i < 0 \\ \textit{i-tá proměnná je nedefinována, jestliže } (\vec{x})_i = 0 \end{cases}$$

kde M_F je incidenční matice (viz str. 14, definice 2.3.4).

Důkaz. „linear autarkies pro $F \subseteq L_F$“: Vezměme lineární autarky v' (částečné ohodnocení splňující podmínku z definice 4.3.2) a k němu příslušnou váhovou funkci w'. Potom vektor

$$(\vec{y})_j = \begin{cases} 0, & \text{jestliže } v'(x_j) \notin var(v') \\ w(x_j), & \text{jestliže } v'(x_j) = 1 \\ -w(x_j), & \text{jestliže } v'(x_j) = 0 \end{cases}$$

splňuje podmínku pro náležení do L_F – pro i-tou složku vektoru vzniklého součinem $M_F \cdot \vec{y}$ totiž platí:

a) pokud autarky nedosazuje do i-té klauzule je součin i-tého řádku M_F s vektorem $\vec{y}$ nulový, protože tento řádek má na všech pozicích, kde má $\vec{y}$ nenulová čísla, hodnotu 0 (značící „proměnná se v klauzuli nevyskytuje“)

b) pokud autarky do i-té klauzule dosazuje, je součin roven $\sum_{j=1}^{n} (M_F)_{i,j} \cdot y_j = \sum_{l \in C_j, v'(l)=1} w'(var(l)) - \sum_{l \in C_j, v'(l)=0} w'(var(l))$, a to je ≥ 0 díky podmínce v definici linear autarkies.

Navíc je vektor $\vec{y} \neq \vec{0}$, protože linear autarky z definice dosazuje alespoň za jednu proměnnou. Interpretací $\vec{y}$ dle uvedených pravidel získáme právě částečné ohodnocení v'.

„linear autarkies pro $F \supseteq L_F$“: pro vektor $\vec{y} \in L_F$ definujeme váhovou funkci $w'(x_i) = (\vec{y})_i$ a částečné ohodnocení v' jako interpretaci $\vec{y}$ dle pravidel uvedených v dokazovaném tvrzení. Ukážeme, že tato volba zajistí splnění nerovnosti

z definice 4.3.2: vezměme libovolnou klauzuli C_i z formule F; díky $\vec{y} \in L_F$ je součin i-tého řádku M_F (odpovídajícího klauzuli C_i) s $\vec{y}$ větší nebo roven 0, tj. $0 \leq \sum_{j=1}^{n} (M_F)_{i,j} \cdot y_j$. Tento součet můžeme rozdělit na splněné a nesplněné literály klauzule C_i a literály, které se v klauzuli C_i nevyskytují (ty odpovídají součinu $y_j \cdot (M_F)_{i,j} = 0$). Dostaneme tak: $0 \leq \sum_{v'(l_j)=1} (M_F)_{i,j} \cdot y_j - \sum_{v'(l_j)=0} (M_F)_{i,j} \cdot y_j$, z čehož již plyne nerovnost požadovaná definicí linear autarkies. (pozn. splněné literály jsou ty, kde $(M_F)_{i,j} \cdot y_j > 0$). □

Tvrzení 4.3.6. [14] *Lze polynomiálně rozhodovat, zda existuje linear autarky. V případě že ano, lze ji v polynomiálním čase najít.*

Důkaz. Podle předchozího tvrzení lineární autarky existuje, právě když je množina $L_F \neq \varnothing$. To nastává právě když a) soustava $M_F \cdot \vec{y} = \vec{0}$ má netriviální řešení (což lze zjistit např. Gaussovou eliminací) nebo b) některá z úloh lineárního programování $M_F \times \vec{y} \geq \vec{e_i}$ má řešení ($\vec{e_i}$ je i-tý vektor kanonické báze) – zde používáme jednoduchého faktu, že množina L_F je uzavřená na násobek kladným racionálním číslem. □

Algoritmus 9 LinAut

Vstup: Formule F v CNF na proměnných $x_1, x_2, \ldots, x_n$ bez jednotkových (a prázdných) klauzulí

Výstup: Splnující pravdivostní ohodnocení proměnných z F, „nesplnitená“, nebo „není LinAut“

```
1:  while F není prázdná množina klauzulí do
2:    M_F := incidenční matice F
3:    if Existuje řešení y⃗ ≠ 0⃗ úlohy M_F · y⃗ ≥ 0⃗ then
4:      t' := Interpretace(y⃗)
5:      F := {C ∈ F : C neobsahuje proměnnou z var(t')}
6:      t := t ∪ t'
7:    else if F je kvadratická formule then
8:      „nesplnitelná“
9:    else
10:     return „není LinAut“
11:   end if
12: end while
13: return t
```

Definice 4.3.7. Formule F je ve třídě LinAut, jestliže uvedený algoritmus skončí úplným dosazením, nebo (nesplnitelnou) kvadratickou formulí bez linear autarkies.

Poznámka 4.3.8. Zde narážíme na problém nastíněný v úvodu: v článku [14] byla definována třída $\mathcal{LSAT}$ tak, že obsahovala pouze klauzule, které lze splnit pomocí postupné aplikace linear autarkies (odpovídala by naší definici po vynechání řádků 7 a 8).

V námi uvedené podobě se třída vyskytla v článku [20]. Přidanou podmínkou jeho autor zajistil, že se v třídě objevily i nesplnitelné formule (a z třídy formulí q-Horn s alespoň kvadratickými klauzulemi se stala vlastní podtřída). Sluší se poznamenat, že podmínka na řádku 7 mohla být uvedena i pro jinou třídu takovou, že

je rozpoznatelná a postupná aplikace linear autarkies pro každou splnitelnou formuli této třídy vede ke splňujícímu ohodnocení.

Problém s jednotkovými klauzulemi je možné vyřešit pomocí spouštění jednotkové rezoluce před během celého algoritmu. Tím by se do třídy LinAut dostaly všechny q-Horn formule (protože třída q-Horn je uzavřená na částečné dosazení a odstraněním klauzule dojde pouze k uvolnění podmínek v definici linear autarky) a všechny kvadratické formule. Ovšem tento předvýpočet by znamenal například to, že by vzniklá třída nebyla uzavřená na odstranění klauzule. Abychom jednomu pojmu nepřiřazovali další význam, rozhodli jsme se držet se definice z článku [20].

Pozorování 4.3.9. *Třída LinAut je uzavřená na odebrání klauzule z formule.*

Důkaz. Odebrání klauzule ubere podmínku pro vlastnoust „bytí linear autarky“. Pokud odebíraná klauzule byla splněna nějakou linear autarky, tak v kroku použití této autarky taktéž zmizela z formule (a uvedená linear autarky zůstává linear autarky i pro formuli, ve které tato klauzule není už od začátku).

Pokud klauzule po výpočtu nad původní formulí zůstala ve formuli, byla formule po výpočtu buďto kvadratická nesplnitelná (po odebrání klauzule se může změnit na splnitelnou, ale ne vypadnout ze třídy), nebo nebyla ve třídě LinAut (po odebrání tam mohla vstoupit, ale to nám nevadí). □

Pozorování 4.3.10. *Třída LinAut je uzavřená na komplementaci proměnné.*

Důkaz. Komplementace proměnné nemá vliv na vlastnost „býti linear autarky“ (protože nerovnost v definici 4.3.2 pracuje s literály), ani na splnitelnost kvadratické formule. □

Pozorování 4.3.11. *Třída LinAut není uzavřená na (obecné) částečné dosazení.*

Důkaz. Nechť F je libovolná formule, která není v LinAut (příklad takové formule bude v tvrzení 4.3.21) a x je proměnná, která se v F nevyskytuje. Do každé klauzule přidáme literál x.

Částečné ohodnocení $v(x) = 1$ splní každou klauzuli (tj. celou formuli) a navíc je linear autarky – stačí volit váhovou funkci tak, že proměnné x přiřadí váhu 1 a zbylým proměnným váhu 0. Pokud ale použijeme ohodnocení $v(x) = 0$, získáme formuli F, která není v LinAut. □

Tvrzení 4.3.12. *Třída LinAut není uzavřená na odebrání literálu.*

Důkaz. Kdyby byla uzavřena na odebrání literálu (právě dokazujeme, že není) i klauzule (což je, viz pozotování 4.3.9), pak by byla uzavřena i na částečné dosazení (což není, viz pozorování 4.3.11) □

Pozorování 4.3.13. *Třída LinAut není uzavřena na spojení dvou formulí.*

Důkaz. Vezměme formule $F_1 = (x_1 \vee x_2 \vee x)\&(\neg x_1 \vee x_2 \vee x)\&(x_1 \vee \neg x_2 \vee x)\&(\neg x_1 \vee \neg x_2 \vee x)$ a $F_2 = (x_1 \vee x_2 \vee \neg x)\&(\neg x_1 \vee x_2 \vee \neg x)\&(x_1 \vee \neg x_2 \vee \neg x)\&(\neg x_1 \vee \neg x_2 \vee \neg x)$. Obě jsou ve třídě LinAut – splní je linear autarky $v_1(x) = 1$, resp. $v_2(x) = 0$, podobně jako v předchozím pozorování.

Nicméně jejich spojením získáme úplnou formuli na třech proměnných, která je nesplnitelná a nemá ani linear autarky (dokonce ani obecnou autarky – kdybychom dosadili za libovolnou(é) proměnnou(é), dosazujeme do všech klauzulí a jelikož autarky musí splnit každou klauzuli, do které dosazuje — to nelze protože je celá formule nesplnitelná). Formule není ani kvadratická. Nepatří tedy do třídy LinAut. □

Tvrzení 4.3.14. *Třída LinAut není uzavřená na komplementaci literálu.*

Důkaz. Vezměme libovolnou formuli F. Posloupností komplementací literálů ji lze převést na formuli F', která obsahuje pouze pozitivní literály. Ta je určitě ve třídě LinAut (váhová funkce zvolená jako identická nula zajistí splnění nerovnosti v definici linear autarky). Kdyby třída LinAut byla uzavřená na komplementaci, můžeme z F' opačným postupem získat F a ta by musela být v LinAut. Třída LinAut však neobsahuje všechny formule (jak je vidět z předchozího tvrzení), což je spor (F byla libovolná). □

Lemma 4.3.15 (o jednoznačnosti dekompozice)**.** [14] (jiný důkaz) *Pro každou formuli F je jednoznačně určená podformule N_{la} (podmnožina klauzulí), pro kterou již neexistuje další linear autarky a lze ji získat z F libovolnou posloupností aplikací linear autarky.*

Důkaz. Nechť F_a a F_b jsou dvě různé podformule (podmnožiny množiny klauzulí) F bez linear autarky, k nimž lze dospět z F postupnou aplikací linear autarkies. Položme $F_c = (F_a \setminus F_b) \cup (F_b \setminus F_a)$. Na F_a ale lze aplikovat stejnou posloupnost linear autarkies, pomocí které jsme z F získali F_b (příp. vynecháme ohodnocování proměnných, které už ve formuli nejsou). Že se stále jedná o linear autarkies je zřejmé – viz důkaz uzavřenosti na odebrání klauzule. To dává spor s tím, že F_a už nemá linear autarky. Symetrickou úvahu lze provést pro F_b. Celkem dostáváme, že množina F_c je prázdná (tj. $F_a = F_b$). □

Poznámka 4.3.16. Formule je splnitelná LinAut, právě když je N_{la} prázdná množina klauzulí. Z toho plyne, že třída LinAut je polynomiálně rozpoznatelná: stačí spustit výše uvedený algoritmus (vzhledem k lemmatu o jednoznačnosti dekompozice je formule N_{la} určena jednoznačně a tedy výsledek nezávisí na pořadí použitých autarkies).

Tvrzení 4.3.17. [14] *Každá ryze kvadratická formule F patří do třídy LinAut.*

Důkaz. V případě, že F je splnitelná, stačí jako váhovou funkci w volit konstantní funkci - každé proměnné přiřadit totéž kladné racionální číslo. Ta bude splňovat podmínku z definice linear autarky pro libovolné splňující ohodnocení formule F – a to tím pádem bude linear autarky (může být nalezeno přímo, nebo postupně přes více linear autarkies).

V případě, že je nesplnitelná, může existovat linear autarky. V tom případě ji algoritmus použije, čímž z formule zmizí některé klauzule, ale formule se nestane splnitelnou (původní formule nebyla splnitelná – neexistovalo splňující ohodnocení, to se nezmění tím, že navíc zafixuji hodnoty některých proměnných). Tím, že aplikuji všechny linear autarkies, které algoritmus najde, získám formuli N_{la}, pro niž již neexistuje linear autarky (viz lemma 4.3.15 o jednoznačnosti dekompozice) a je nesplnitelná (a samozřejmě kvadratická). Algoritmus (v uvedené verzi) použije podmínku z řádku 7. □

Následující lemma a jeho důsledek pocházejí z článku [20].

Lemma 4.3.18. [20] *Předpokládejme, že formule F má complexity index Z a nemá linear autarky. Potom platí jedna z následujících podmínek:*

1. *Pro nějakou klauzuli i (délky L_i) je $L_i < 2Z$*

2. Všechny klauzule mají délku $L = 2Z$

Důkaz. Jestliže neplatí 1, pak zřejmě $L - 2Z\vec{e} \geq 0$. Tedy z definice complexity indexu $M_F \cdot \vec{x} \geq L - 2Z\vec{e} \geq 0$ pro nějaké $\vec{x} \in [-1, 1]^n$. Protože neexistuje linear autarky, je $\vec{x} = \vec{0}$, což znamená, že $L = 2Z\vec{e}$. □

Důsledek 4.3.19. [20] *Každá q-Horn formule bez jednotkových klauzulí patří do třídy LinAut.*

Důkaz. Nejdříve připomeňme, že třída q-Hornovských formulí je uzavřená na odebrání klauzule, tedy i na použití linear autarky.

Nyní pro spor předpokládejme, že máme formuli F (rozumí se neprázdnou), která je q-Horn bez jednotkových klauzulí, nemá linear autarky (to můžeme bez újmy na obecnosti předpokládat dle výše uvedeného) a není ve třídě LinAut. Z toho, že je formule q-Hornovská víme, že má complexity index $Z \leq 1$. Z předchozího lemmatu dostáváme, že pro délky klauzulí L_i platí:

1. $1 \geq Z > \frac{L_i}{2}$ pro nějakou klauzuli i, nebo
2. $1 \geq Z = \frac{L_i}{2}$ pro každou klauzuli i

přičemž případ 1 nemůže nastat (taková klauzule by byla jednotková). Plátí tedy 2, to znamená, že jsou všechny klauzule kvadratické a taková formule patří do třídy LinAut dle tvrzení 4.3.17, což je spor. □

Tvrzení 4.3.20. [14] *Třída LinAut obsahuje třídu Matched formulí.*

Důkaz. Matched formule mají vždy počet klauzulí ≤ počet proměnných – to znamená, že incidenční matice M_F má alespoň tolik sloupců, kolik řádků. Díky tomu pro Matched formuli vždy existuje linear autarky (využijeme myšlenek z důkazu tvrzení 4.3.6):

a) jestliže má matice hodnost n (tzn. je čtvercová regulární), získáme linear autarky interpretací řešení rovnosti $M_F \cdot \vec{x} = \vec{e_1}$,

b) v opačném případě generuje soustava $M_F \cdot \vec{x} = \vec{0}$ alespoň jednodimenzionální prostor řešení a tedy lze vybrat řešení netriviální.

Nyní si všimněme, že rozdělením klauzulí matched formule do libovolných dvou množin budou obě vzniklé formule opět matched. Po každé aplikaci linear autarky tak získáme opět matched formuli (protože aplikace autarky odstraňuje celé klauzule), která má linear autarky. „Nesplnitelné jádro“ N_{la} z lemmatu o jednoznačnosti dekompozice je tak prázdná formule, to znamená, že pro matched formuli najdeme pomocí linear autarkies úplné splňující ohodnocení (je LinAut). □

Tvrzení 4.3.21. *Pro každé i je třída LinAut neporovnatelná se SLUR(i).*

Důkaz. $LinAut \not\subseteq SLUR(i)$: Plyne z toho, že třída LinAut obsahuje třídu q-Horn formulí bez jednotkových klauzulí, která je se SLUR(i) neporovnatelná pro každé i (viz důkaz tvrzení 4.2.11).

$SLUR(i) \not\subseteq LinAut$: Víme, že platí SLUR $\subsetneq$ SLUR(i). Stačí tedy najít SLUR formuli, která nebude ve třídě LinAut. Takovou formuli najdeme v [20]: $F = (\neg x_1 \vee$

$x_2 \vee x_3)\&(x_1 \vee \neg x_2 \vee x_3)\&(x_1 \vee x_2 \vee \neg x_3)\&(\neg x_1 \vee \neg x_2 \vee x_3)$. Tato formule není LinAut, protože jediné řešení $M_F \cdot \vec{x} \geq \vec{0}$ je $\vec{x} = \vec{0}$. M_F má pro tuto formuli následující tvar:

$$M_F = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

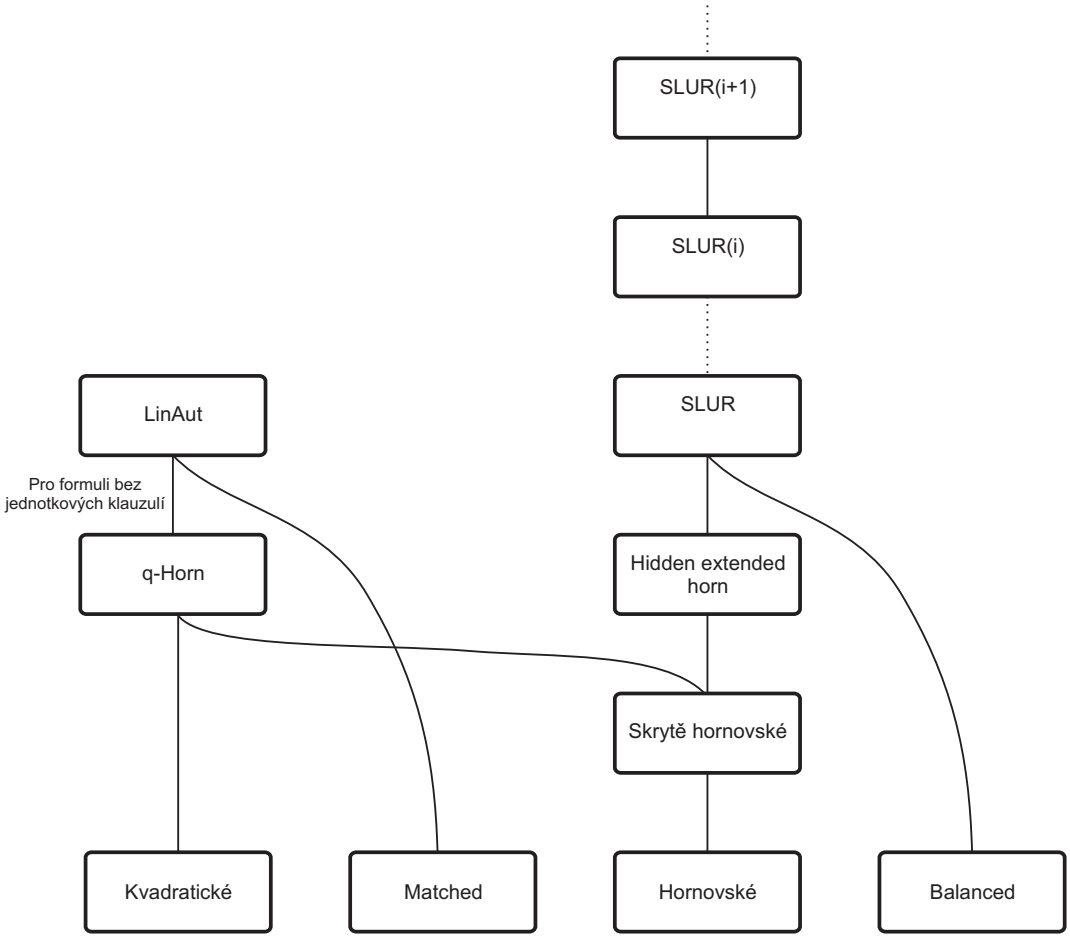
To, že je formule SLUR, lze nahlédnout z toho, že je splněná, právě když je sudý počet proměnných ohodnocen 1 (tedy algoritmus SLUR zajistí splnění správným ohodnocením poslední proměnné). □

4.4 Shrnutí vlastností vztahů vzhledem k inkluzi

Následující tabulka shrnuje vlastnosti zkoumaných tříd co do uzavřenosti na některé operace s formulemi.

Třída	Odebrání		Komplementace		Částečné dosazení	Spojení formulí
	literálu	klauzule	literálu	proměnné		
Kvadratické	ano	ano	ano	ano	ano	ano
Balanced	ne	ano	ne	ano	ano	ne
Matched	ne	ano	ano	ano	ne	ne
Hornovské	ano	ano	ne	ne	ano	ano
Skrytě horn.	ano	ano	ne	ano	ano	ano
Hid. ext. horn	ne	ano	ne	ano	ano	ne
SLUR	ne	ne	ne	ano	ano	ne
q-Horn	ano	ano	ne	ano	ano	ne
LinAut	ne	ano	ne	ano	ne	ne

Zbývá už jen diagram inkluzí:



Obrázek 4.2: Diagram inkluzí

Literatura

[1] Arora S., Barak B.: *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, 2009.

[2] Aspvall B.: *Recognition disguised NR(1) instances of satisfiability problem*, J. Algorithms 1 (1980), 97–103.

[3] Aspvall B., Plass M. F., Tarjan R. E.: *A linear-time algorithm for testing the truth of certain quantified boolean formulas*, Inform. Process. Lett., Vol. 8, Issue 3 (1979), 121–123.

[4] Boros E., Hammer P. L., Sun X.: *Recognition of q-Horn formulae in linear time*, Discrete Appl. Math., Vol. 55, Issue 1 (1994), 1–13.

[5] Boros E., Crama Y., Hammer P. L., Saks M.: *A complexity index for satisfiability problems*, SIAM J. Comput., Vol. 23, Issue 1 (1994), 45–49.

[6] Conforti M., Cornuéjols G., Vušković K.: *Balanced Matrices*, Discrete Mathematics, Vol. 306, Issue 19–20, 2006, 2411–2437.

[7] Crama Y., Hammer P. L.: *Boolean Function: Theory, Algorithms and Applications*. Draft knihy je dostupný na internetové adrese http://www.rogp.hec.ulg.ac.be/Crama/Publications/BookPage.html

[8] Čepek O., Kučera P.: *Known and new classes of generalized Horn formulae with polynomial recognition and SAT testing*

[9] Davis M., Putnam H.: *A computing Procedure for Quantification Theory*, J. Assoc. Comput. Mach., Vol. 7, No. 3, 1960, 201-215.

[10] del Val A.: *On 2-SAT and Renamable Horn*, Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, July 30-August 03 2000, 279–284.

[11] Franco J. V., van Gelder A.: *A perspective on certain polynomial-time solvable classes of satisfiability*, Discrete Appl. Math., Vol. 125, Issue 2–3 (2003), 177–214.

[12] Chandrasekaran R.: *Integer programming problems for which a simple rounding type of algorithm works*. V W. R. Pulleyblabnk (ed.) *Progress in Combinatorial Optimization*, Academic Press Canada, Toronto, Ontario, Kanada, 1984, 101–106.

[13] Chandru V., Hooker J. N.: *Extended Horn sets in propositional logic*, J. Assoc. Comput. Mach., Vol. 38, No. 1, 1991, 205–221.

[14] Kullmann O.: *Investigation on autark assignment*, Discrete Appl. Math., Vol. 107, 2000, 99–137.

[15] Matoušek J., Gärtner B.: *Understanding and Using Linear Programming*. Springer, Berlin Heidelberg, 2007.

[16] Schlipf J. S., Annexstein F. S., Franco J. V., Swaminathan R. P.: *On Finding Solutions for the Extended Horn Formulas*, Inform. Process. Lett., Vol. 54, Issue 3 (1995), 133–137.

[17] Tarjan, R.: *Depth-first search and linear graph algorithms*, SIAM J. Comput., Vol. 1, No. 2 (1972), 146–160.

[18] Trümper K.: *Effective Logic Computation*. Wiley, New York, 1998.

[19] Trümper K.: *Monotone decomposition of matrices*. Technical report UTDCS–1–94, Universiti of Texas in Dallas, 1994.

[20] van Maaren H.: *A short Note on Linear Autarkies, q-Horn Formulas and the Complexity Index*, DIMACS Technical Report 99-26, 1999.