Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Maroš Hrnčiar

DPLL Algoritmus a Výrokové Důkazy

Katedra algebry

Vedoucí bakalářské práce: prof. RNDr. Jan Krajíček, DrSc.

Studijní program: Matematika

Studijní obor: Matematické metody informační bezpečnosti

Praha 2012

Za neoceniteľné pripomienky, rady a poskytnutú literatúru by som sa na tomto mieste rád poďakoval vedúcemu mojej práce, prof. RNDr. Janovi Krajíčkovi, DrSc., za jazykovú a štylistickú korektúru Michalovi Mutňanskému.

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne a výhradne s použitím citovaných prameňov, literatúry a ďalších odborných zdrojov.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona v platnom znení, najmä skutočnosť, že Univerzita Karlova v Praze má právo na uzavretie licenčnej zmluvy o použití tejto práce ako školského diela podľa §60 odst. 1 autorského zákona.

V Prahe dňa 25.5.2012 Maroš Hrnčiar

Názov práce: DPLL Algoritmus a Výrokové Důkazy

Autor: Maroš Hrnčiar

Katedra: Katedra algebry

Vedúci bakalárskej práce: prof. RNDr. Jan Krajíček, DrSc., Katedra algebry

Abstrakt: Dôkazová zložitosť je zaujímavá súčasť matematiky nachádzajúca sa na pomedzí obrovskej oblasti logiky a teórie zložitosti. Skúma aké dôkazové systémy sú potrebné na efektívne dokazovanie rôznych matematických tvrdení. Predmetom tejto práce je spojenie medzi dôkazovými systémami a algoritmami na SAT. Uvidíme, že beh algoritmu na nesplniteľnej formule môže byť nahliadnutý ako výrokový dôkaz jej nesplniteľnosti, čím samotný algoritmus prakticky definuje celý dôkazový systém. Práca je určená najmä čitateľom so záujmom o dôkazovú zložitosť, ale dokáže aj samostatne objasniť princíp rezolúcie, či ponúknuť menej obvyklý pohľad na SAT, no zároveň predpokladá čitateľovu znalosť základov výrokovej logiky, teórie grafov a zložitosti.

Kľúčové slová: splnitelné formuly, SAT, rezolúcia

Title: DPLL algorithm and propositional proofs

Author: Maroš Hrnčiar

Department: Katedra algebry

Supervisor: prof. RNDr. Jan Krajíček, DrSc., Katedra algebry

Abstract: Proof complexity is an interesting mathematical part connecting logic and complexity theory. It investigates which proof systems are needed for effective theorem proving. The aim of this paper is to present a relation between propositional proof systems and SAT algorithms. We will see that a run of an algorithm on the unrealizable formula can be seen as a propositional proof of its unsatisfiability, so the algorithm practically defines whole proof system. The thesis is mainly recommended for readers interested in proof complexity, but it can also independently illustrate a resolution principle and perhaps show some less common view of SAT assuming reader's basic knowledge of propositional logic, graph theory and complexity.

Keywords: satisfiable formulas, SAT, resolution

Obsah

Úvod

Systémy na dokazovanie výrokových tautológií a algoritmy na *SAT* sú dve zdanlivo odlišné veci. V tejto práci nájdete okrem podrobného teoretického úvodu k jednotlivým oblastiam ilustrovanom na jednoduchých príkladoch aj odpoveď na otázku, čo ich spája a ako veľmi spolu v skutočnosti súvisia. Podkladom definícií a potrebnej teórie je z veľkej časti práca [6], kde autor okrem iného hovorí o *SAT* ako o jednom z najštudovanejších algoritmických problémov počítačovej vedy vôbec. Je fakt, že v posledných troch desaťročiach veľká časť výskumu bola smerovaná k pochopeniu jeho matematickej štruktúry a vyvýjaniu nových algoritmov. Teória dôkazovej zložitosti je zasa vnímaná od vydania pôvodného článku *Cooka* a *Reckhowa* [4], v ktorom matematicky zadefinovali výrokový dôkaz ako taký, zaviedli a zovšeobecnili pojem dôkazového systému a skúmali vzťahy medzi veľkosťami dôkazov a triedami zložitosti.

Táto práca je rozčlenená do piatich kapitol. V *prvej* čitateľa oboznámime so základnými pojmami, v ktorých sa budeme po celý čas pohybovať a popíšeme niektoré všeobecne známe fakty skúmanej oblasti, či jednoduché tvrdenia. *Kapitola 2* predstaví základný úplný a korektný algoritmus na *SAT* a následne uvedie nejaké jeho vylepšenia. Oproti tomu *tretia kapitola* opisuje príklady úplných a korektných dôkazových systémov výrokových tautológii a analyzuje jednotlivé typy stromových dôkazov. V *kapitole 4* sa konečne odhalí a dokáže vzájomný vzťah týchto dvoch prístupov akoby základný cieľ celej práce. Avšak nájsť tautológie kandidujúce na to byť "ťažkými"vôbec nie je jednoduchá úloha. Preto v *záverečnej kapitole* jeden netriviálny príklad odhalíme, pozrieme sa na jeho zložitosť, skonštruujeme výrokový dôkaz, nájdeme vhodnú heuristiku pre algoritmy a budeme pozorovať silu dokázaných tvrdení z predchádzajúcich sekcií na zaujímavom príklade.

Obrázky a grafy použité v práci boli vytvorené programom *AutoCAD 2010.*

1. Vo svete výrokovej logiky

„Pokiaľ pripustíme jeden nezmysel, ostatné už dokážeme z neho...“

Citujúc *Aristotela*, samotného zakladateľa, vás vítam vo svete axiómov, pravidiel a zároveň záhad, či už dokázaných, nedokázaných, alebo takých, o ktorých je dokázané, že sú nedokázateľné, nech už to znie akokoľvek absurdne, vo svete logiky.

No logika ako súčasť matematiky začala byť vnímaná až o viac ako dve tisícročia neskôr, od polovice 19.storočia, kedy jej tento nový rozmer dal anglický matematik a filozof *George Boole*, po ktorom je aj pomenovaná jedna z jej častí, **Booleovská (výroková) logika**.

Výroková logika študuje formy usudzovania, pre ktoré platnosť záverov nezávisí od obsahu ani vnútornej štruktúry výrokov, ale výlučne len na ich pravdivosti, či nepravdivosti.

1.1 Výroková formula

Booleovská funkcia n premenných je ľubovolná funkcia $f : \{0,1\}^n \to \{0,1\}$, kde $\{0,1\}^n$ značí množinu všetkých usporiadaných n-tíc z čísel $0,1$.

Booleovská formula (ďalej len „formula“) je reťazec reprezentujúci booleovskú funkciu, obsahujúci:

- výrokové premenné: $p_1, p_2, ...$
- základné operácie:
  - nulárne: konštanty 0,1
  - unárnu: negácia ($\neg$)
  - binárne: konjunkcia ($\wedge$), disjunkcia ($\vee$), implikácia ($\to$)
- pomocné symboly (napr. zátvorky)

Definujeme ju nasledovne:

1. Každá premenná je formula.
2. Konštanty $0,1$ sú formuly.
3. Ak A je formula, potom $\neg A$ je formula.

4. Ak A, B sú formuly, $(A \wedge B), (A \vee B), (A \rightarrow B)$ sú takisto formuly.

5. Konečne veľa aplikácií bodov (1)-(4) vytvára formulu.

Pravdivostné ohodnotenie α je každé zobrazenie $\alpha : D(\alpha) \rightarrow \{0, 1\}$, kde $D(\alpha)$ je neprázdna množina niektorých výrokových premenných. Ohodnotenie α sa nazýva **úplné** pre formulu A, ak $Var(A) \subseteq D(\alpha)$, pričom $Var(A)$ je množina všetkých premenných vyskytujúcich sa vo formule A. Pokiaľ situácia $Var(A) \subseteq D(\alpha)$ nenastáva, hovoríme o **čiastočnom** ohodnotení formuly A.

Každé ohodnotenie α vieme rozšíriť na funkciu α' definovanú na množine formúl vybudovaných z $D(\alpha)$, ktorá priradí takej formule číslo $a \in \{0, 1\}$ nasledovným rekurzívnym spôsobom:
Pre všetky formuly B, C: $Var(B), Var(C) \subseteq D(\alpha)$

- $\alpha'(\neg B) := 1 - \alpha(B)$
- $\alpha'(B \wedge C) := \alpha(B) \cdot \alpha(C)$
- $\alpha'(B \vee C) := 1 - ((1 - \alpha(B)) \cdot (1 - \alpha(C)))$

Funkciu α' s ohodnotením α spravidla stotožňujeme.

Nech A, B sú formuly. A a B sú **ekvivalentné** ($A \equiv B$), ak $\alpha(A) = \alpha(B)$ pre každé α, ktoré je úplné pre A aj B.

Formula A sa nazýva **tautológia**, ak je pravdivá vo všetkých úplných pravdivostných ohodnoteniach α pre A, t.j. $\alpha(A) = 1$ pre všetky α úplné, napr. $A = (x \vee \neg x)$.
Formula A sa nazýva **nesplniteľná**, ak je nepravdivá vo všetkých úplných pravdivostných ohodnoteniach α pre A, t.j. $\alpha(A) = 0$ pre každé α úplné, napr. $A = (x \wedge \neg x)$.

Formula A je **splniteľná**, ak existuje pravdivostné ohodnotenie $\alpha : \alpha(A) = 1$. Takémuto pravdivostnému ohodnoteniu potom hovoríme **splňujúce ohodnotenie** A.

Problém splniteľnosti (SAT), rozhodovací problém zaoberajúci sa splniteľnosťou booleovských výrazov, rieši otázku, či daná formula je splniteľná. Je to historicky prvý NP-úplný problém (*Cook-Levinova veta*, [3]), teda s vysokou pravdepodobnosťou neexistuje polynomiálny algoritmus, ktorý SAT rieši.

Veta 1. *SAT je algoritmicky rozhodnuteľný.*

Dôkaz. Nech A je formula. Potom počet premenných, ktoré A obsahuje, je konečné číslo, t.j. $|Var(A)| = n$. Teda existuje 2^n rôznych úplných ohodnotení pre A:

$\alpha_1, ..., \alpha_{2^n}$. Postupným skúšaním jednotlivých $\alpha_i, i \in \{1, ..., 2^n\}$ a spočítaním $\alpha_i(A)$ z rekurzívnych vzťahov overíme, či A je splniteľná. □

Dôsledok 2. *TAUT (rozhodovací problém riešiaci, či formula je tautológia) je algoritmicky rozhodnuteľný.*

Dôkaz. Nech A je formula. A je tautológia práve vtedy, keď pre každé úplné ohodnotenie α platí $\alpha(A) = 1$. To nastane práve vtedy, keď pre každé $\alpha : \alpha(\neg A) = 0$, čo je definícia nesplniteľnej formuly $\neg A$. Teda $TAUT$ je rozhodnuteľný triviálnym algoritmom z dôkazu pre SAT. □

Literálom nazývame takú formulu l, ktorá sa rovná premennej x alebo jej negácii.

Disjunkcii konečného množstva literálov hovoríme **klauzula**, t.j. C je klauzula práve vtedy, keď existuje prirodzené číslo k a literály $l_1, l_2, ..., l_k$ tak, že $C = l_1 \vee l_2 \vee ... \vee l_k$. Špeciálnym prípadom klauzuly je prázdna klauzula, ktorá sa označuje $\emptyset$.

Formula A je v **disjunktívnej normálnej forme (DNF)**, keď je disjunkciou termov, t.j. podformulí, z ktorých každá je konjunkciou konečného množstva literálov, čo znamená, že existujú formuly $E_1, E_2, ..., E_m$ také, že $A = \bigvee_{j=1}^{m} E_j$ a súčasne $E_i = l_1^i \wedge l_2^i \wedge ... \wedge l_{k_i}^i$ pre každé $i \in \{1, 2, ..., m\}$.

Formula A je v **konjunktívnej normálnej forme (CNF)**, keď je konjunkciou klauzúl, teda existujú klauzuly $C_1, C_2, ..., C_m$ také, že $A = \bigwedge_{j=1}^{m} C_j$ pre m > 0.

Pozorovanie 3. *CNF-formula je tautológia práve vtedy, keď každá jej klauzula je tautológia.*

Dôkaz. Nech C je CNF formula zložená z klauzúl $C_1, C_2, ..., C_m$.
"$\Rightarrow$": Pre spor predpokladajme, že existuje také $j \in \{1, 2, ..., m\}$, že C_j nie je tautológia. Potom musí existovať nejaké ohodnotenie α, pri ktorom $\alpha(C_j) = 0$. Aplikovaním rekurzívneho pravidla na konjunkciu však musí platiť aj $\alpha(C) = 0$, čím sme našli ohodnotenie, pri ktorom CNF formula C nie je splniteľná a teda zároveň spor s predpokladom, že C je tautológia.
"$\Leftarrow$": Nech α je ľubovolné pravdivostné ohodnotenie premenných v C. Užitím rekurzívneho pravidla pre konjunkciu priamo z predpokladu, že pre všetky $i \in \{1, 2, ..., m\}$ platí $\alpha(C_i) = 1$ plynie, že $\alpha(C) = 1$. □

Tvrdenie 4. *Ku každej formule A existuje formula B v $CNF(DNF)$ - tvare taká, že $A \equiv B$.*

Dôkaz je možné zostrojiť opakovaným použitím *De Morganových zákonov*, viď. [12].

Nech C je klauzula, l_i jej literály také, že $l_i = x_i$ alebo $l_i = \neg x_i$, $i \in \{1, 2, ..., m\}$ a α ohodnotenie niektorých premenných x_i.

Reštrikcia C pod α je klauzula

$$C|_\alpha = \begin{cases} 1 & \text{ak } \exists (l_j) \in C : \alpha(l_j) = 1 \\ \emptyset & \text{ak } \alpha(l_j) = 0 \ \forall (l_j) \in C \\ \bigvee_j l_j : x_j \notin D(\alpha) & \text{inak} \end{cases}$$

Nech $A = \bigwedge_{k=1}^{n} C_k$ je CNF formula, α ohodnotenie niektorých premenných $x_i \in Var(A)$, $J_\alpha \subseteq \{1, .., n\}$ množina takých $j \in \{1, .., n\}$, že $C_j|_\alpha$ je rôzna od $\emptyset$, 1 a obsahuje nejaký neohodnotený literál.

Reštrikcia A pod α je formula definovaná nasledovne:

$$A|_\alpha = \begin{cases} 0 & \text{ak } \exists\ j \in \{1, .., n\} : C_j|_\alpha = \emptyset \\ 1 & \text{ak } C_j|_\alpha = 1 \ \forall j \in \{1, .., n\} \\ \bigwedge_{j \in J_\alpha} C_j|_\alpha & \text{inak} \end{cases}$$

Lemma 5. *Nech A je CNF formula, α ohodnotenie a x premenná vyskytujúca sa vo $Var(A)$, $x \notin D(\alpha)$. Ďalej nech β je úplné ohodnotenie pre A, $\alpha \subseteq \beta$. Potom:*

1. *β je splňujúce ohodnotenie $A \Leftrightarrow \beta$ je splňujúce ohodnotenie $A|_\alpha$.*
2. *$A|_\alpha$ je splniteľná $\Leftrightarrow A|_{\alpha \cup \{x:=1\}}$ alebo $A|_{\alpha \cup \{x:=0\}}$ je splniteľná.*

Dôkaz.

1. Stačí ukázať, že $\beta(C|_\alpha) = \beta(C)$ pre všetky $C \in A$. $\beta(C) = 1 \Leftrightarrow \exists l \in C :\ \beta(l) = 1 \Leftrightarrow \exists l \in C :\ \alpha(l) = 1$ alebo $(\beta - \alpha)(l) = 1 \Leftrightarrow \beta(C|_\alpha) = 1$.
2. "$\Rightarrow$": Nech β je splňujúce ohodnotenie $A|_\alpha$, $\alpha \subseteq \beta$. Potom (z časti (1)) β je splňujúce ohodnotenie A. Nakoľko pre nejaké $e \in \{0, 1\}$ musí byť $\alpha \cup \{x := e\} \subseteq \beta$, potom β musí byť splňujúce ohodnotenie buď $A|_{\alpha \cup \{x:=0\}}$, alebo $A|_{\alpha \cup \{x:=1\}}$.
   "$\Leftarrow$": Nech $\alpha' = \alpha \cup \{x := e\}$ a β je splňujúce ohodnotenie $A|_{\alpha'}$ pre $e \in \{0, 1\}$. Bez ujmy na všeobecnosti predpokladajme, že $\alpha \subseteq \beta$. Položme $\beta' := (\beta - \beta(x)) \cup \{x := e\}$. Potom $\alpha \subseteq \alpha' \subseteq \beta'$ a pretože $x \notin Var(A|_{\alpha'})$, β' je splňujúce ohodnotenie $A|_{\alpha'}$. Teraz už stačí dva krát aplikovať (1) a dostávame najskôr, že β' je splňujúce ohodnotenie A a následne aj splňujúce ohodnotenie $A|_\alpha$.

$\square$

1.2 Dôkazový systém

Výrokový dôkazový systém (DS) je každá polynomiálne spočítateľná funkcia $P : \{0,1\}^* \to \{0,1\}^*$, ktorej oborom hodnôt je množina všetkých výrokových tautológií ($Rng(P) = TAUT$).

Každé slovo $w \in \{0,1\}^*$ také, že $P(w) = \tau$ sa nazýva **P-dôkaz** tautológie τ. Zložitosť τ , značená $C_P(\tau)$, je veľkosť najmenšieho takého w.

Z daného P-dôkazu je ľahké určiť, ktorá formula je ním dokázaná a overiť korektnosť tohoto dôkazu. Avšak generovať dôkazy pre danú formulu je už zložité a tieto dôkazy môžu byť veľmi dlhé v porovnaní s veľkosťou formuly.

Jedným z najjednoduchších príkladov DS je nasledovne definovaná funkcia f:
Ak $w = (v, A)$, kde v je tabuľka pravdivostných hodnôt formuly A, ktorá má v poslednom stĺpci odpovedajúcom formule A samé jednotky, potom $f(w) := A$, inak $f(w) := 1$.
Pri n premenných danej formuly však tabuľka musí mať presne 2^n riadkov a $n+1$ stĺpcov, čiže w je exponenciálne veľký dôkaz vzhľadom k veľkosti formuly a ako príklad nezaujímavý.

Dôkazové systémy možno porovnávať. Slúži na to pojem simulácia. Majme 2 DS: P a S. S **simuluje** P ($P \leq S$), keď existuje polynóm p taký, že pre každú tautológiu τ a P-dôkazy π formuly τ existuje S-dôkaz π' formuly τ taký, že $|\pi'| \leq p(|\pi|)$. Ak $P \leq S$, hovoríme, že S je aspoň taký silný ako P.

$P \approx S$ (P a S sú **ekvivalentné**), ak $P \leq S$ a súčasne $S \leq P$. $\approx$ je relácia ekvivalencie.

Dôkazový systém je **optimálny** [10], ak simuluje všetky ostatné. Existencia optimálneho dôkazového systému je zatiaľ otvorený problém.

DS P je **polynomiálne ohraničený**, ak existuje polynóm p taký, že pre každú tautológiu τ existuje P-dôkaz w veľkosti $|w| \leq p(|\tau|)$. T.j. dôkaz je maximálne polynomiálne dlhý vzhľadom k dĺžke formuly.

Veta 6. *(Cook-Reckhow, [4]) Polynomiálne ohraničený dôkazový systém existuje práve vtedy, keď $NP = coNP$.*

Dôkaz. "$\Leftarrow$": Rozhodovací problém $TAUT$, ktorý rieši otázku, či daná formula je tautológia, je z Cookovej vety $coNP$-úplný. Dokážeme, že ak existuje výrokový dôkazový systém P s uvedenou vlastnosťou, tak potom $TAUT \in NP$, z čoho vyplynie $NP = coNP$.
Majme teda takýto polynomiálne ohraničený dôkazový systém P. Nedeterministický Turingov stroj akceptujúci $TAUT$ bude fungovať nasledovne:

- Dostane τ, o ktorej chce povedať, či je tautológia
- Natipuje dôkaz w dĺžky $\leq p(|\tau|)$.
- Deterministicky overí, či $P(w) = \tau$.

"$\Rightarrow$": Nech $NP = coNP$. Potom $TAUT \in NP$. Nech M je nedeterministický Turingov stroj akceptujúci $TAUT$. Výrokový dôkazový systém P, v ktorom má každá tautológia polynomiálny dôkaz, potom možno definovať nasledovne:

$$P(w) = \begin{cases} \tau & \text{ak } w \text{ je akceptačný výpočet stroja } M \text{ na } \tau \\ 1 & \text{inak} \end{cases}$$

$\square$

Dôsledok 7. *Ak neexistuje polynomiálne ohraničený DS, potom $P \neq NP$.*

2. DPLL algoritmus

2.1 Motivácia

Ako bolo uvedené v predošlej kapitole, zistiť, či formula F s n premennými je splniteľná, je relatívne jednoduché preskúmaním všetkých 2^n možných úplných ohodnotení α_i a overením, či $F|_{\alpha_i} = 1$. Avšak v čase $O(|F| \cdot 2^n)$.

Nasledujúcim vylepšením sa pokúsime eliminovať niekoľko ohodnotení, ktoré budú triviálne nesplňujúce, teda nebudú sa musieť skúšať vôbec. Budeme ich však vytvárať postupne a pri každom ohodnotení nejakého literálu, t.j. vytvorení čiastočného ohodnotenia α skontrolujeme, či $F|_\alpha$ nie je už splnená, alebo nesplnená.

Dokonca ak by $F|_\alpha = 1$, každé úplné ohodnotenie $\beta \supseteq \alpha$ bude podľa *Lemmy 5* splniteľné pre F.
Pokiaľ by $F|_\alpha = 0$, potom je možné vynechať z testovania všetky ohodnotenia, v ktorých by premenné nadobúdali rovnaké hodnoty ako nadobúdajú v α.

2.2 Popis

DPLL algoritmus je algoritmus slúžiaci na rozhodovanie splniteľnosti výrokových formúl založený na vyššie zmienenom jednoduchom triku s postupným vytváraním čiastočných ohodnotení a rekurzívnom prehľadávaní do hĺbky. Uvedený bol v roku 1962 pánmi *M. Davisom*, *H. Putnamom*, *G. Logemannom* a *D.W. Lovelandom* (odtiaľ názov *DPLL*) ako zúplnenie už 2 roky predtým známej metódy na overovanie splniteľných formúl.

Algoritmus na začiatku zistí, či daná CNF formula F je splniteľná alebo nesplniteľná triviálne (neobsahuje žiadnu klauzulu, respektíve obsahuje prázdnu klauzulu, spor). Potom vyberie premennú x_i a aplikuje algoritmus rekurzívne na podformulu získanú z pôvodnej formuly dosadením hodnoty 0 alebo 1 do x_i. Keď zistí, že F je splniteľná, vráti splňujúce ohodnotenie spätným chodom rekurzie, inak aplikuje algoritmus na formulu získanú dosadením druhej hodnoty. Ak ani tu neuspeje, F je nesplniteľná.

Pseudokód algoritmu $DPLL$ (F, α) :
VSTUP: F, α, kde F je formula v CNF tvare, α jej čiastočné ohodnotenie
VÝSTUP: splňujúce ohodnotenie F, prípadne $\spadesuit$, ak je F nesplniteľná

1. IF $F|_\alpha = 0$ THEN RETURN $\spadesuit$
2. IF $F|_\alpha = 1$ THEN RETURN α
3. Zvoľ premennú $x \in F|_\alpha$ a $a \in \{0, 1\}$
4. $\beta := DPLL(F, \alpha \cup \{x := a\})$
5. IF $\beta \neq \spadesuit$ THEN RETURN β
   ELSE RETURN $DPLL(F, \alpha \cup \{x := (1 - a)\})$

Voľba premennej v kroku 3 na algoritmus vplyv má, no pre začiatok uvažujeme voľbu ľubovoľnú.

Tvrdenie 8. *Nech F je formula. $DPLL(F, \emptyset)$ vždy skončí a vráti buď splňujúce ohodnotenie F, pokiaľ je F splniteľná, alebo $\spadesuit$, ak je nesplniteľná.*

Dôkaz. F je formula, α čiastočné ohodnotenie, $n = |Var(F|_\alpha)|$. Indukciou cez n ukážeme, že $DPLL(F, \alpha)$ skončí a vráti $\alpha' \supseteq \alpha$ splňujúce ohodnotenie F, pokiaľ $F|_\alpha$ splniteľná, respektíve $\spadesuit$ v opačnom prípade.

$n = 0 \Rightarrow F|_\alpha = 0$ alebo $F|_\alpha = 1$, čím algoritmus skončí v prvých dvoch krokoch. V prvom prípade je $F|_\alpha$ nesplniteľná, algoritmus vráti $\spadesuit$, v druhom prípade splniteľná a algoritmus vráti α.

$n > 0 \Rightarrow$ Nech $x \in Var(F|_\alpha)$ a $\alpha_i = \alpha \cup \{x := i\}$, $i \in \{0, 1\}$. $F|_{\alpha_i}$ obsahuje najviac $n - 1$ premenných.

Ak $F|_\alpha$ je nesplniteľná, potom $F|_{\alpha_i}$ podľa *Lemmy 5* je nesplniteľná tiež pre ľubovoľné $i \in \{0, 1\}$. Preto v tomto prípade $DPLL(F, \alpha_i)$ z indukčného predpokladu skončí a vráti $\spadesuit$ pre každé i. A teda aj $DPLL(F, \alpha)$ skončí a vráti $\spadesuit$.

Teraz predpokladajme, že $F|_\alpha$ je splniteľná. Potom z *Lemmy 5* $F|_{\alpha_i}$ je splniteľná pre $i = 0$ alebo $i = 1$.

Ak $F|_{\alpha_0}$ splniteľná, potom z indukčného predpokladu výkon $DPLL(F, \alpha_0)$ skončí a vráti čiastočné splňujúce ohodnotenie, a teda taktiež aj $DPLL(F, \alpha)$.

Pokiaľ $F|_{\alpha_0}$ nesplniteľná, výkon $DPLL(F, \alpha_0)$ skončí a vráti $\spadesuit$. Naviac $F|_{\alpha_1}$ musí byť splniteľná, preto z indukčného predpokladu výkon $DPLL(F, \alpha_1)$ vráti čiastočné splňujúce ohodnotenie, a teda rovnako aj $DPLL(F, \alpha)$. $\square$

Dôsledok 9. *Nech F je formula. Ak $DPLL(F, \emptyset)$ vráti nejaké splňujúce ohodnotenie F, potom F je splniteľná. Pokiaľ vráti $\spadesuit$, F je nesplniteľná.*

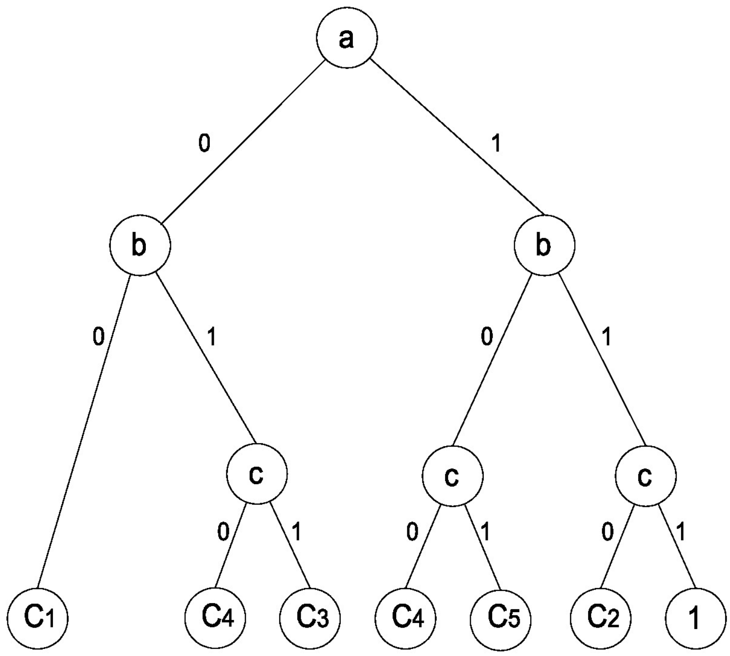
Týmto sme ukázali, že $DPLL$ algoritmus je úplný a korektný algoritmus na rozhodovanie splniteľnosti formúl v CNF tvare.

2.3 Reprezentácia

Realizácia $DPLL$ algoritmu na formule F býva zväčša reprezentovaná zakoreneným binárnym stromom $T(F)$. Uzly stromu predstavujú jednotlivé rekurzívne volania. Označujeme ich, s výnimkou listov, premennými, za ktoré práve v algoritme dosadzujeme. Z vnútorných uzlov vždy vedú 2 hrany, možné ohodnotenia premennej, ktorou je uzol označený. Pre každý uzol U, cesta z koreňa do U definuje čiastočné ohodnotenie premenných α a rekurzívne volanie na tento uzol je aplikované na podformulu $F|_\alpha$ získanú reštrikciou pôvodnej formuly. Cesty z koreňa do listov predstavujú jednotlivé ohodnotenia.

Ak je nejaký list stromu vyhodnotený ako pravdivý pre nejaké ohodnotenie *(success leaf)*, formula je splniteľná. $DPLL$ algoritmus určí, či takýto list existuje, vytýči cestu od neho ku koreňu, čím nájde úplné splňujúce ohodnotenie premenných a v strome ho označí jednotkou. Ak taký list neexistuje, formula je nesplniteľná (pre každé úplné ohodnotenie α existuje klauzula C vo formule F taká, že $C|_\alpha = \emptyset$). Tieto listy *(failure leafs)* značíme buď nulou, prípadne počiatočnými klauzulami, ktoré spôsobili nesplnenosť pri ohodnotení korešpondujúcim s cestou od koreňa k danému *failure* listu.

Nech F je formula $\overbrace{(a \vee b)} \wedge \overbrace{(\neg a \vee \neg b \vee c)} \wedge \overbrace{(a \vee \neg b \vee \neg c)} \wedge \overbrace{(c)} \wedge \overbrace{(\neg a \vee b \vee \neg c)}$ s klauzulami $C_1, ..., C_5$.



Obr. 2.1: *Príklad stromu reprezentujúceho $DPLL$ algoritmus na formule F*

2.4 Vylepšenia

Samotný $DPLL$ algoritmus, v podobe v akej je uvedený, však stále nerieši problém s efektívnosťou a môže sa zdať, že jeho konštrukcia bola zbytočná. No časová náročnosť sa dá rapídne zlepšiť použitím vhodne zvolených orezávacích techník a heuristík, ktoré sa aplikujú pri každom volaní algoritmu. Je to napríklad vhodné poradie premenných, ktoré sa idú vyhodnocovať (ako prvé voliť také, ktoré sa vyskytujú v krátkych klauzulách, respektíve také, ktoré sú vo formule veľakrát), či vhodné zvolenie $a \in \{0, 1\}$.

Základnými ľahko implementovateľnými pravidlami [11] na zefektívnenie $DPLL$ algoritmu sú:

1. **Pravidlo jednotkového literálu**:
   Ak v klauzule sú všetky literály, až na jeden, ohodnotené nejakým čiastočným ohodnotením ako nepravdivé, prípadne klauzula obsahuje len jeden literál, potom tento jediný zostávajúci literál musí nadobúdať hodnotu 1. Nová CNF formula, ktorá vznikne aplikovaním tohto pravidla na nejakú jej klauzulu, má rovnakú splnitelnosť ako pôvodná.

2. **Pravidlo unipolárneho literálu**:
   Ak sa v CNF formule vyskytuje nejaký literál l_i iba v jednej fáze, t.j. buď l_i alebo $\neg l_i$, nie obidva súčasne, potom všetky klauzuly obsahujúce tento literál môžeme z formuly odstrániť alebo danému literálu priradiť hodnotu 1. Novovzniknutá formula opäť zachováva splniteľnosť pôvodnej.

3. **Pravidlo konfliktu**:
   Ak nejaké čiastočné ohodnotenie zapríčini nesplniteľnosť klauzuly, teda všetky jej literály budú ohodnotené ako nepravdivé, potom v podstrome pod týmto čiastočným ohodnotením nemôžeme nájsť splňujúce ohodnotenie, všetky listy budú *failure*. Pravidlo je realizované pridaním klauzuly C_α k formule $F|_\alpha$. $C_\alpha = \{\neg x | \alpha(x) = 1\} \cup \{x | \alpha(x) = 0\}$, t.j. C_α je najväčšia možná klauzula nesplnená ohodnotením α. Platí: $F \equiv F_\alpha \Leftrightarrow F_\alpha$ je nesplniteľná, a teda algoritmus nikdy nezačne hľadať splňujúce ohodnotenie v podstrome určenom α: $F_\alpha = 0$.

4. **Pravidlo pravdivého literálu**:
   Ak nejaké čiastočné ohodnotenie zapríčini splniteľnosť klauzuly, t.j. aspoň jeden literál bude ohodnotený ako pravdivý, potom klauzulu môžeme z formuly odstrániť. Splniteľnosť zostane zachovaná.

5. **Pravidlo tautologickej klauzuly**:
   Ak nejaká klauzula obsahuje premennú p_i a súčasne premennú $\neg p_i$, obsahuje podformulu 1 a preto ju môžeme odstrániť za zachovania splniteľnosti, nakoľko formula 1 je splnená zakaždým.

Príklad:
Určite splniteľnosť formuly

$$F = \overbrace{(a \vee b)} \wedge \overbrace{(b \vee c)} \wedge \overbrace{(\neg b \vee c \vee d)} \wedge \overbrace{(\neg a \vee e \vee a)} \wedge \overbrace{(a \vee \neg e)} \wedge \overbrace{(\neg a \vee \neg d)} \wedge$$
$$\wedge \overbrace{(\neg b \vee \neg d)} \wedge \overbrace{(a \vee \neg c \vee d)} \wedge \overbrace{(\neg e \vee d)} \wedge \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)}.$$

Riešenie:
Jednotlivé klauzuly označíme $C_1, ..., C_{12}$ a postupujeme vylepšeným $DPLL$ algoritmom:

1. C_4 obsahuje premennú a a súčasne $\neg a$, t.j. je tautologickou klauzulou a podľa $P5$ ju nemusíme uvažovať.
   $F_1 = C_1 \wedge C_2 \wedge C_3 \wedge C_5 \wedge ... \wedge C_{12}$

2. Premenná e sa v F_1 vyskytuje len v literáloch tvaru $\neg e$ a podľa $P2$ môžeme z F_1 odstrániť aj klauzuly C_5 a C_9.
   $F_2 = C_1 \wedge C_2 \wedge C_3 \wedge C_6 \wedge C_7 \wedge C_8 \wedge C_{10} \wedge C_{11} \wedge C_{12}$

3. Zvolíme premennú $a \in F_2$.

4. Uvažujme $F_2|_{a=0}$:

   (a) Klauzula C_1 spĺňa predpoklady $P1$, teda nutne $b = 1$.

   (b) Použitím $P4$ môžeme zasa odstrániť C_6.
   $$F_3 = \overbrace{(1)} \wedge \overbrace{(1 \vee c)} \wedge \overbrace{(0 \vee c \vee d)} \wedge \overbrace{(0 \vee \neg d)} \wedge \overbrace{(0 \vee \neg c \vee d)} \wedge$$
   $$\wedge \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)}$$

   (c) Kvôli $P4$ môžeme z F_3 odstrániť klauzuly C_1, C_2.
   $$F_4 = \overbrace{(0 \vee c \vee d)} \wedge \overbrace{(0 \vee \neg d)} \wedge \overbrace{(0 \vee \neg c \vee d)} \wedge \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)}$$

   (d) Následným použitím $P1$ na klauzulu C_7 musí byť $d = 0$.
   $$F_5 = \overbrace{(0 \vee c \vee 0)} \wedge \overbrace{(0 \vee 1)} \wedge \overbrace{(0 \vee \neg c \vee 0)} \wedge \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)}$$

   (e) Po ďalšom aplikovaní $P1$, tentokrát na klauzulu C_3 dostávame
   $$F_6 = \overbrace{(0 \vee 1 \vee 0)} \wedge \overbrace{(0 \vee 1)} \wedge \overbrace{(0 \vee 0 \vee 0)} \wedge \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)},$$
   čím sme zapríčinili nesplniteľnosť klauzuly C_8 a teda podľa $P3$ pre $F_2|_{a=0}$ neexistuje žiadne splňujúce ohodnotenie.

5. Teda uvažujme $F_2|_{a=1}$:

   (a) Podľa $P4$ odstránime klauzuly C_1, C_8 a z $P1$ pre C_6 vyplýva, že $d = 0$.
   T.j. $F_7 = \overbrace{(b \vee c)} \wedge \overbrace{(\neg b \vee c \vee 0)} \wedge \overbrace{(0 \vee 1)} \wedge \overbrace{(\neg b \vee 1)} \wedge$
   $\wedge \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)}$.

   (b) Použitím $P4$ môžeme odstrániť C_6, C_7 a aplikovaním $P2$ s premennou c takisto C_2, C_3. Dostávame $F_8 = \overbrace{(f \vee g)} \wedge \overbrace{(\neg f \vee g)} \wedge \overbrace{(f \vee \neg g)}$.

   (c) Zvolíme ďalšiu premennú $f \in F_8$.

   (d) Uvažujme $F_8|_{f=0}$:
   $F_9 = \overbrace{(0 \vee g)} \wedge \overbrace{(1 \vee g)} \wedge \overbrace{(0 \vee \neg g)}$
   Z $P1$ na klauzule C_{10} musí platiť $g = 1$, avšak tým pádom C_{12} je nesplniteľná klauzula a teda aj $F_8|_{f=0}$ je nesplniteľná.

   (e) Uvažujme $F_8|_{f=1}$:
   $F_{10} = \overbrace{(1 \vee g)} \wedge \overbrace{(0 \vee g)} \wedge \overbrace{(1 \vee \neg g)}$
   Z $P1$ na klauzulu C_{11} tento raz vyplýva, že $g = 1$, čím dostávame
   $F_{11} = \overbrace{(1 \vee 1)} \wedge \overbrace{(0 \vee 1)} \wedge \overbrace{(1 \vee 0)} = 1$ a teda máme splňujúce ohodnotenie pre F, t.j. F je splniteľná.

3. Rezolúcia

Rezolučná metóda (R) je výrokový dôkazový systém, ktorý dokazuje, že daná DNF formula je tautológia. Je založený na zrejmom pozorovaní, že formula A je tautológia práve vtedy, keď formula $\neg A$ je nesplniteľná. Z nesplniteľných formúl v CNF tvare (A DNF práve vtedy, keď $\neg A$ CNF) bude možné touto metódou odvodiť prázdnu klauzulu, spor, pomocou rezolučného odvodzovacieho pravidla, jediného odvodzovacieho pravidla, ktoré rezolúcia používa.

3.1 Všeobecná rezolúcia

3.1.1 Rezolučné pravidlo

A, B klauzuly, l literál. Z formuly $(A \vee \{l\}) \wedge (B \vee \{\neg l\})$ odvoď $(A \vee B)$.

Lemma 10. *Rezolučné pravidlo je korektné, t.j. ak A, B klauzuly, α je splňujúce ohodnotenie formuly $(A \vee \{l\}) \wedge (B \vee \{\neg l\})$, potom α je takisto splňujúce ohodnotenie $(A \vee B)$.*

Dôkaz. Nech α je splňujúce ohodnotenie $(A \vee \{l\}) \wedge (B \vee \{\neg l\})$. Potom α musí byť splňujúce ohodnotenie jednotlivých klauzúl $(A \vee \{l\})$ aj $(B \vee \{\neg l\})$.

Ďalej môžu nastať 3 prípady:

1. Ak $\alpha(l) = 0$, potom musí existovať literál $k \in A$ taký, že $\alpha(k) = 1$ a teda $\alpha(A) = 1$.
2. Pokiaľ $\alpha(l) = 1$, potom nutne $\alpha(\neg l) = 0$ a z rovnakého dôvodu ako v prípade 1 musí byť $\alpha(B) = 1$.
3. Prípad, že $l \notin D(\alpha)$ je jasný, pretože vtedy s určitosťou existujú literály $m \in A$, $n \in B$ také, že $\alpha(m) = 1$ a $\alpha(n) = 1$ a teda $\alpha(A) = 1$ aj $\alpha(B) = 1$.

Všetky prípady vedú k tomu, že buď $\alpha(A) = 1$ alebo $\alpha(B) = 1$, no potom musí platiť aj $\alpha(A \vee B) = 1$, čo sme chceli ukázať. $\square$

3.1.2 Rezolučný dôkaz

Nech A je DNF formula tvaru $A = \bigvee_{i \in I} B_i : B_i = \bigwedge_{j \in J_i} l^i_j$. Definujme $C_i := \neg B_i = \bigvee_{j \in J_i} \neg l^i_j \ \forall i \in I$, teda $\neg A = \bigwedge_{i \in I} C_i$ je CNF formula.

Rezolučný dôkaz tautológie A definujeme ako postupnosť klauzúl $D_1, D_2, ..., D_t$ takých, že

1. D_u je jedna z C_i, alebo je odvodená rezolučným pravidlom z D_{v_1}, D_{v_2} pre nejaké $v_1, v_2 < u$ $\forall u \in \{1, 2, ..., t\}$

2. D_t je prázdna klauzula, spor

Príklad:
Dokážte, že formula
$\neg F = \overbrace{(\neg x \wedge \neg y \wedge m)} \vee \overbrace{(\neg m \wedge \neg z)} \vee \overbrace{(x \wedge \neg t)} \vee \overbrace{(\neg x \wedge y)} \vee \overbrace{(t)} \vee \overbrace{(z)}$
je tautológia.

Riešenie:
$F = \overbrace{(x \vee y \vee \neg m)} \wedge \overbrace{(m \vee z)} \wedge \overbrace{(\neg x \vee t)} \wedge \overbrace{(x \vee \neg y)} \wedge \overbrace{(\neg t)} \wedge \overbrace{(\neg z)}$

Označme klauzuly v CNF formule F postupne $D_1, ..., D_6$.
Opakovaným použitím rezolučného pravidla zostrojíme rezolučný dôkaz $D_1, ..., D_{11}$:

1. $(x \vee y \vee z) =: D_7 \longleftarrow^m D_1, D_2$

2. $(y \vee z \vee t) =: D_8 \longleftarrow^x D_7, D_3$

3. $(x \vee z \vee t) =: D_9 \longleftarrow^y D_8, D_4$

4. $(t \vee z \vee t) =: D_{10} \longleftarrow^x D_9, D_3$

5. $(z) =: D_{11} \longleftarrow^t D_{10}, D_5$

6. $\emptyset =: D_{12} \longleftarrow^z D_{11}, D_6$

Veta 11. *Ak $D_1, D_2, ..., D_t$ je rezolučný dôkaz formuly A, potom je $\neg A$ nesplniteľná.*

Dôkaz. Pre spor predpokladajme, že existuje α, nejaké splňujúce ohodnotenie $\neg A = \bigwedge\limits_{i \in I} C_i$. Potom α je zároveň splňujúce ohodnotenie klauzúl C_i, $\forall i \in I$.

α tým splĺňa aj všetky klauzuly D_j, ktoré sa nachádzajú medzi počiatočnými $C_i, i \in I$.

Ostatné klauzuly D_k daného rezolučného dôkazu musia byť už odvodené z tých, ktoré sme práve označili za splniteľné ohodnotením α a podľa *Lemmy 10* je tým pádom α splňujúce ohodnotenie aj týchto ostatných klauzúl.

V takom prípade však α splĺňa aj D_t, čo je prázdna klauzula, z definície nesplniteľná. Dostávame sa do sporu, čím sme dokázali korektnosť R. $\square$

3.1.3 Reprezentácia

Rezolučné dôkazy sa zvyčajne znázorňujú ako orientované grafy bez cyklov, kde každý vrchol je označený jednou z klauzúl $D_1, D_2, ..., D_t$ a medzi vrcholmi (D_i, D_k) a súčasne (D_j, D_k) existujú hrany práve vtedy, keď D_k je odvodená z D_i, D_j rezolučným pravidlom ($\frac{D_i \quad D_j}{D_k}$).

V grafe dôkazu nejakej formuly jednotlivé hrany označujeme literálmi, na ktoré je práve rezolučné pravidlo uplatňované, t.j. tými, ktoré sa práve eliminujú, napríklad pri ($\frac{A=A'\cup\{x\} \quad B=B'\cup\{\neg x\}}{C=A'\cup B'}$) by boli hrany $(A, C), (B, C)$ označené literálmi x, respektíve $\neg x$.

3.2 Stromová rezolúcia

Pokiaľ v dôkaze π tautológie A každá už raz rezolučne odvodená klauzula D_i je použitá pre odvodenie ďalšej klauzuly D_j najviac jeden raz, potom grafom takéhoto dôkazu je strom a dôkazu π hovoríme **stromový**.
Dôkazový systém pripúšťajúci len stromové dôkazy sa nazýva **stromová rezolúcia** (R^*).

3.2.1 Rezolučný strom

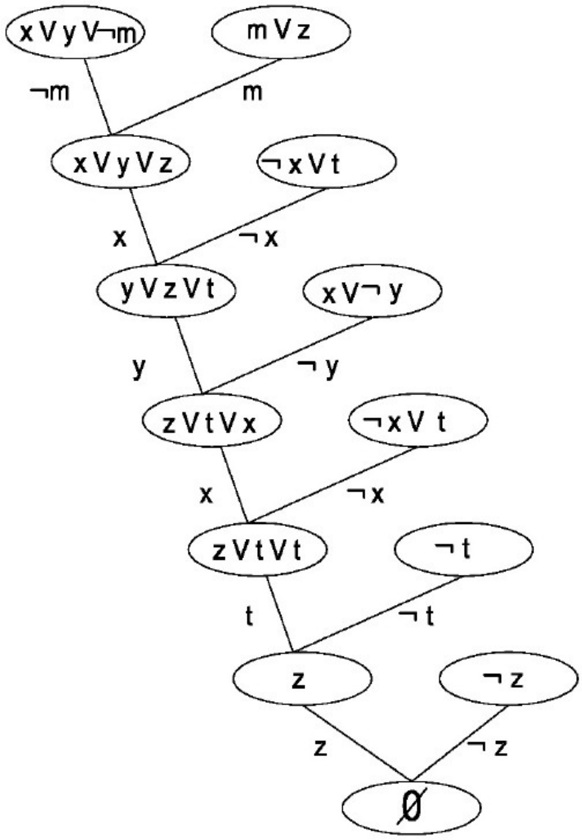
Nech F je CNF formula (budeme ju stotožňovať s množinou jej klauzúl), C klauzula a T binárny strom, v ktorom sú vrcholy označené klauzulami a hrany premennými. T je rezolučný strom pre C z F, ak:

1. koreň T je označený klauzulou C
2. každý list T je označený klauzulou $E \in F$
3. ak vnútorný uzol je označený nejakou klauzulou D a má dve deti (2 susediace uzly smerom ďalej od koreňa) označené klauzulami D_0, D_1, potom D je odvodená z D_0, D_1 rezolučným pravidlom vylúčením premennej x a hrany (D_0, D), (D_1, D) sú označené literálmi x a $\neg x$, prípadne naopak, v závislosti na príslušnosti k jednotlivým klauzulám D_0, D_1.

Veľkosťou stromu T rozumieme počet jeho vrcholov. Vrcholy stromu a ich označenia pre jednoduchosť ďalej stotožňujeme.

Tautológia A v DNF tvare je teda dokázaná v R^* len vtedy, keď existuje T, rezolučný strom pre prázdnu klauzulu z CNF formuly $\neg A$.

Nakoľko je v polynomiálnom čase možné overiť, či klauzuly D_0, D_1 sú odvodené z D rezolučným pravidlom, je takisto spočítateľné v polynomiálnom čase, či T je rezolučný strom pre prázdnu klauzulu z danej CNF formuly, a teda R^* je dobre definovaný dôkazový systém.



Obr. 3.1: *Rezolučný strom pre prázdnu klauzulu z formuly F z predošlého príkladu*

Lemma 12. *Nech F je CNF formula, T rezolučný strom pre C z F. Ak F je splnená ohodnotením α, potom C je splnená ohodnotením α ($F|_\alpha = 1 \Rightarrow C|_\alpha = 1$).*

Dôkaz. Indukciou cez hĺbku rezolučného stromu pre C z F:

Ak C je list, potom $C \in F$ a $C|_\alpha = 1$ plynie z $F|_\alpha = 1$ ihneď.

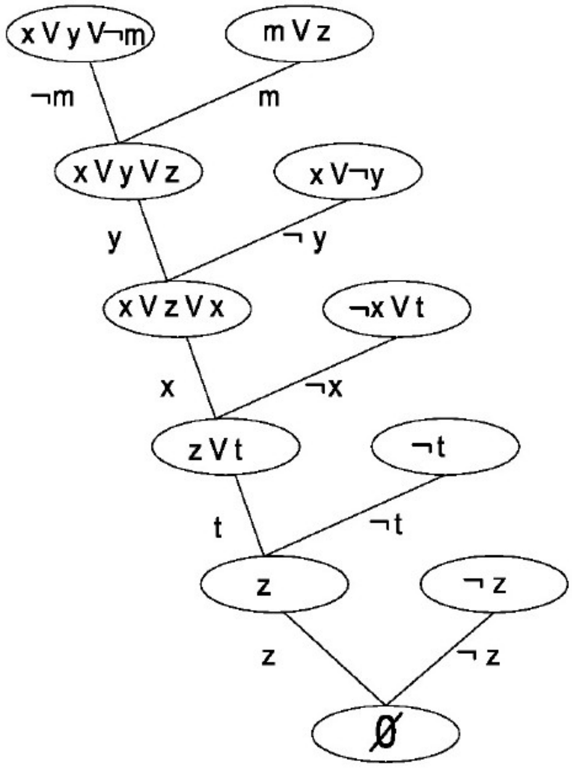
Inak, vrchol C musí mať dve deti C_0, C_1, pričom C je odvodený z C_0 a C_1 rezolučným pravidlom eliminovaním nejakej premennej. Rezolučné stromy pre C_0 a C_1 z F majú hĺbku menšiu ako je hĺbka rezolučného stromu pre C z F, a teda z indukčného predpokladu $C_0|_\alpha = C_1|_\alpha = 1$. No z *Lemmy 10* následne plynie aj $C|_\alpha = 1$. □

Dôsledok 13. *Ak F je CNF formula, T rezolučný strom pre prázdnu klauzulu z F, potom F je nesplniteľná.*

Dôkaz. Keby α bolo splňujúce ohodnotenie F, $F|_\alpha = 1$, podľa *Lemmy 12* by α bolo splňujúce ohodnotenie pre prázdnu klauzulu z F, ktorá je však z definície nesplniteľná. □

3.2.2 Regulárna stromová rezolúcia

Nech T je rezolučný strom. T je **x-regulárny**, ak každá cesta z koreňa do listu obsahuje najviac jednu hranu označenú premennou x alebo $\neg x$. T je **regulárny**, ak je x-regulárny pre každú premennú x označujúcu niektorú z jeho hrán.
Dôkaz, ktorého grafom je regulárny strom sa nazýva **regulárny stromový** a dôkazový systém pripúšťajúci len takéto dôkazy značíme $regR^*$.



Obr. 3.2: *Regulárny rezolučný strom pre prázdnu klauzulu z formuly F z posledného príkladu*

3.2.3 Vzťah R* a regR*

Veta 14. *Nech F je CNF formula. Ak existuje rezolučný strom pre prázdnu klauzulu z F, potom existuje aj regulárny rezolučný strom pre prázdnu klauzulu z F.*

Dôkaz. Nech F je CNF formula a T rezolučný strom veľkosti s pre prázdnu klauzulu z F. Dôkaz indukciou podľa s: Ak $s = 1$, $Var(T) = \emptyset$ a samotný T je už regulárny.

Ďalej nech $x \in Var(T)$ je premenná, pre ktorú platí, že T nie je x-regulárny (pokiaľ taká neexistuje, T už regulárny je a nič netreba dokazovať). Označme d počet hrán stromu T, ktoré sú označené buď x alebo $\neg x$ takých, že na ceste k prázdnej klauzule existuje aj nejaká iná hrana označená x alebo $\neg x$.

Potom $d > 0$ a nech e je jedna takáto hrana, ktorá súčasne je aj jednou z hrán vyskytujúcich sa pri aplikácii rezolučného pravidla na klauzuly C_0, D_0 a odvodení klauzuly C_1. Majme tiež postupnosť klauzúl $C_0, C_1, ..., C_n = \emptyset$ nachádzajúcich sa na ceste z e do prázdnej klauzuly. Označme e' prvú hranu rôznu od e na tejto ceste, ktorá je označená x alebo $\neg x$ a korešponduje s aplikáciou rezolučného pravidla na klauzuly C_k, D_k.

Bez ujmy na všeobecnosti predpokladajme, že $C_1 = (C_0 \setminus \{x\}) \cup (D_0 \setminus \{\neg x\})$, $C_{k+1} = (C_k \setminus \{x\}) \cup (D_k \setminus \{\neg x\})$ a skonštruujme rezolučný strom T' pre prázdnu klauzulu z F veľkosti menšej ako s.

Ako prvé z T odstránime klauzulu D_0 spolu so všetkými jej predchodkyňami a umiestnime $C_1^{'} := C_0$ spolu s celým podstromom T_{C_0} namiesto C_1. Ďalej za predpokladu, že $C_{i+1} = (C_i \setminus \{x_i^{\epsilon}\}) \cup (D_i \setminus \{x_i^{1-\epsilon}\})$, $\epsilon \in \{0, 1\}$, nasledovným rekurentným spôsobom vytvoríme ostatné klauzuly $C_i^{'}$, $i \in \{2, ..., n\}$ aby $C_{i+1}^{'} \subseteq C_{i+1} \cup \{x\}$ pre $i < k$ a $C_{i+1}^{'} \subseteq C_{i+1}$ pre $i \geq k$. Ak $x_i^{\epsilon} \in C_i^{'}$, potom $C_{i+1}^{'} := (C_i^{'} \setminus \{x_i^{\epsilon}\}) \cup (D_i \setminus \{x_i^{1-\epsilon}\})$ a nahradíme C_{i+1} klauzulou $C_{i+1}^{'}$. V opačnom prípade položíme $C_{i+1}^{'} := C_i^{'}$ a D_i spolu s podstromom T_{D_i} odstránime.

Výsledný pozmenený strom $T^{'}$ je rezolučným stromom, aký sme chceli vytvoriť a keďže má o minimálne dve klauzuly menej ako mal T, t.j. $s^{'} = |T^{'}| < s$, z indukčného predpokladu vyplýva, že $T^{'}$ je možné transformovať na regulárny rezolučný strom pre prázdnu klauzulu z F. □

Dôsledok 15. $R^* \equiv regR^*$.

Úplnosť R^* bude dokázaná v nasedujúcej kapitole.

4. Súvislosť DPLL algoritmu a stromovej rezolúcie

Beh $DPLL$ algoritmu na nesplniteľnej formule môže byť interpretovaný ako dôkaz jej nesplniteľnosti a pri rezolučných stromoch to platí aj naopak.

4.1 Úplnosť R*

Lemma 16. *Nech F je nesplniteľná formula. Potom $DPLL(F, \alpha)$ vykoná párny počet rekurzívnych volaní pri ľubovolnom ohodnotení α.*

Dôkaz. Indukciou cez počet rekurzívnych volaní $DPLL$ algoritmu: Ak algoritmus nevykoná žiadne rekurzívne volanie, tvrdenie platí, pretože 0 je párne číslo. Predpokladajme teraz, že $DPLL(F, \alpha)$ ich vykoná $r > 0$. Podľa *Tvrdenia 8* algoritmus vráti $\spadesuit$. Z definície však potom existujú volania $DPLL$ algoritmu v bodoch 4 a 5, ktoré vykonajú r_0 a r_1 rekurzívnych volaní. Potom $r_0 + r_1 + 2 = r$, z indukčného predpokladu r_0, r_1 sú párne a teda aj r je párne. $\square$

Lemma 17. *Nech F je CNF formula, α ohodnotenie. Ak $DPLL(F, \alpha)$ vráti $\spadesuit$, potom existuje klauzula C taká, že $C|_\alpha = 0$ a regulárny rezolučný strom T pre C z F, že $Var(T) \cap D(\alpha) = \emptyset$.*

Dôkaz. Majme $DPLL(F, \alpha)$, ktorý vráti $\spadesuit$ po s rekurzívnych volaniach. Podľa *Lemmy 16*, s je párne. Dôkaz indukciou cez s:

Ak $s = 0$, algoritmus nevykoná žiadne rekurzívne volanie a keďže vráti $\spadesuit$, musí existovať klauzula C v F taká, že $C|_\alpha = 0$. Teda rezolučný strom obsahujúci len koreň označený C je taký, ako sme chceli.

Pre indukčný krok predpokladajme, že $s = r + 2$, kde r je párne prirodzené číslo. Nakoľko $DPLL(F, \alpha)$ neskončí v prvých dvoch bodoch, určite existuje nejaká premenná x zvolená v bode 3. Pretože algoritmus vráti $\spadesuit$, existuje volanie $DPLL(F, \alpha_0)$ v bode 4, ktoré vykoná s_0 ďalších rekurzívnych volaní a takisto volanie $DPLL(F, \alpha_1)$ v bode 5, ktoré vykoná s_1 ďalších rekurzívnych volaní, kde $\alpha_i = \alpha \cup \{x := i\}$ a $s = s_0 + s_1 + 2$. Keďže obe tieto volania vrátia $\spadesuit$, indukčný predpoklad tvrdí, že existujú také klauzuly C_0, C_1, že $C_i|_{\alpha_i} = 0$ a C_i má regulárny rezolučný strom T_i, $Var(T_i) \cap D(\alpha_i) = \emptyset$ pre $i = 0$ a $i = 1$.

Ak $x \notin C_0$ alebo $\neg x \notin C_1$, môžeme položiť $T := T_0$, respektíve $T := T_1$ a tvrdenie platí. V opačnom prípade si musíme uvedomiť, že T_0 ani T_1 nemôžu obsahovať hrany označené x, $\neg x$. Preto regulárny rezolučný strom T môže byť odvodený z T_0 a T_1 spojením ich koreňov do nového koreňa $C = (C_0 - \{x\}) \cup (C_1 - \{\neg x\})$ s označením príslušných hrán literálmi x, $\neg x$. Pretože vieme, že $C_i|_{\alpha_i} = 0$ pre $i \in \{0, 1\}$, $\neg x$ nemôže patriť do C_0, ani x do C_1. Teda $C \subseteq (C_0 \cup C_1) - \{x, \neg x\}$, preto $C|_\alpha = 0$ a T je regulárny rezolučný strom pre C z F. □

Veta 18. *(Úplnosť regR*) Pre každú nesplniteľnú CNF formulu F existuje regulárny rezolučný strom pre prázdnu klauzulu z F.*

Dôkaz. Majme nesplniteľnú CNF formulu F. Kvôli korektnosti nám algoritmus $DPLL(F, \emptyset)$ vráti ♠. Pretože prázdna klauzula je jediná taká klauzula C, že $C|_\emptyset = 0$, veta je priamym dôsledkom *Lemmy 17* aplikovaného na formulu F a ohodnotenie $\alpha = \emptyset$. □

Z predošlej vety a *Dôsledku 15* plynie, že R^* je úplný dôkazový systém.

Lemma 19. *Nech F je nesplniteľná CNF formula obsahujúca premenné $x_1, ..., x_n$. Potom existuje nejaká premenná $x_i \in Var(F)$, $i \in \{1, ..., n\}$ taká, že F obsahuje aspoň jednu klauzulu, v ktorej je x_i a nie je $\neg x_i$ a súčasne obsahuje aspoň jednu klauzulu, v ktorej je $\neg x_i$ a nie je x_i.*

Dôkaz. Pre spor nech taká x_i neexistuje, t.j. pre všetky $i \in \{1, ..., n\}$, buď F neobsahuje klauzulu C_k: $x_i \in Var(C_k)$ a $\neg x_i \notin Var(C_k)$, alebo F neobsahuje klauzulu C_l: $\neg x_i \in Var(C_l)$ a $x_i \notin Var(C_l)$. Zostrojíme ohodnotenie α tak, že pre $i \in \{1, ..., n\}$, $x_i := 0$ práve vtedy, keď nastane prvý prípad a $x_i := 1$ keď nastane druhý. Potom však $C_j(\alpha) = 1$ pre každé j a formula F bude splniteľná, čo je spor s predpokladom. □

Alternatívny dôkaz úplnosti $regR^*$ skonštruujeme indukciou podľa počtu premenných.

Dôkaz. Nech F je nesplniteľná CNF formula s klauzulami $C_1, ..., C_k$ obsahujúca premenné $x_1, ..., x_n$.

Ak $n = 1$, potom podľa *Lemmy 19* aspoň jedna z klauzúl formuly F obsahuje literál x_1 a neobsahuje $\neg x_1$ a takisto jedna z klauzúl určite obsahuje $\neg x_1$, pričom zároveň neobsahuje x_1. Bez ujmy na všeobecnosti, nech sú tieto dve klauzuly $C_1 = x_1$, $C_2 = \neg x_1$. Aplikovaním rezolučného pravidla na C_1, C_2 dostávame prázdnu klauzulu a jednoducho môžeme zostrojiť regulárny rezolučný strom.

Nech $n > 1$ a premenná spĺňajúca *Lemmu 19* je napríklad x_n. Klauzuly $C_1, ..., C_k$ rozdelíme do štyroch skupín. Také, čo neobsahujú x_n ani $\neg x_n$ budú tvoriť prvú skupinu, také, čo obsahujú x_n aj $\neg x_n$ druhú skupinu, klauzuly obsahujúce x_n a súčasne neobsahujúce $\neg x_n$ zaradíme do tretej skupiny a zostávajúce klauzuly budú patriť do štvrtej skupiny. Následne zostrojíme CNF formulu F' tak, že bude obsahovať klauzuly z prvej skupiny a všetky možnosti klauzúl vzniknutých aplikovaním rezolučného pravidla na premennú x_n v klauzulách zo skupín 3 a 4.

Predpokladajme, že β je ohodnotenie premenných $x_1, ..., x_n$, ktoré spĺňa F'. Potom však β musí spĺňať všetky klauzuly patriace do prvej skupiny, pretože tie sú zhodné s niektorými klauzulami F' a ďalej buď všetky klauzuly tretej skupiny *(I)*, alebo všetky klauzuly štvrtej skupiny *(II)*. Inak by existovali klauzuly C_u z tretej skupiny a C_v zo štvrtej tak, že všetkým ich literálom okrem $x_n, \neg x_n$ by ohodnotenie β priradilo nulu a teda aplikovaním rezolučného pravidla na tieto dve klauzuly by vznikla nová klauzula K, pre ktorú by $\beta(K) = 0$. To však nastať nemôže, pretože K je klauzula formuly F' a tá je ohodnotením β splnená.

Teraz rozšírime ohodnotenie β na β' spôsobom, že $x_n := 0$ pokiaľ nastane prípad *(I)*, $x_n := 1$ v prípade *(II)*. Tým sme však zostrojili také ohodnotenie, ktoré spĺňa aj všetky klauzuly v tretej a štvrtej skupine.

Nakoľko klauzuly v druhej skupine sú splnené triviálne, dokázali sme, že F' je rovnako nesplniteľná. Obsahuje premenné $x_1, ..., x_{n-1}$, teda z indukčného predpokladu existuje regulárny rezolučný strom pre prázdnu klauzulu z F'. Listy tohoto stromu, ktoré nie sú zároveň pôvodnými klauzulami F, musia byť z nich odvodené rezolučným pravidlom vylúčením premennej x_n, čiže ich doplnením do stromu získavame regulárny rezolučný strom pre prázdnu klauzulu z F. □

Beh $DPLL$ algoritmu na nesplniteľnej CNF formule $\neg A$ vieme reprezentovať binárnym stromom, v ktorom z koreňa a každého vnútorného uzla vedú práve dve hrany. Rekurzívnym spôsobom popísaným v dôkaze *Lemmy 17* aplikovanom na formule $\neg A$ a ohodnotení $\alpha = \emptyset$ dokážeme tento strom transformovať na regulárny rezolučný strom pre prázdnu klauzulu z $\neg A$.

Pozorovanie 20. *Skonštruovaný strom je rezolučný strom nahliadnutý zhora nadol, teda beh $DPLL$ algoritmu na $\neg A$ môžeme chápať ako rezolučný dôkaz tautológie A.*

Za predpokladu použitia niektorých pravidiel na vylepšenie $DPLL$, v grafe reprezentujúcom beh konkrétneho algoritmu môže existovať vnútorný uzol, respektíve koreň, z ktorého vedie iba jedna hrana. V takomto prípade pri transformácii na regulárny rezolučný strom musíme pracovať s ekvivalentnou formulou doplnenou o ďalšie klauzuly zostrojené v popise pravidla 3.

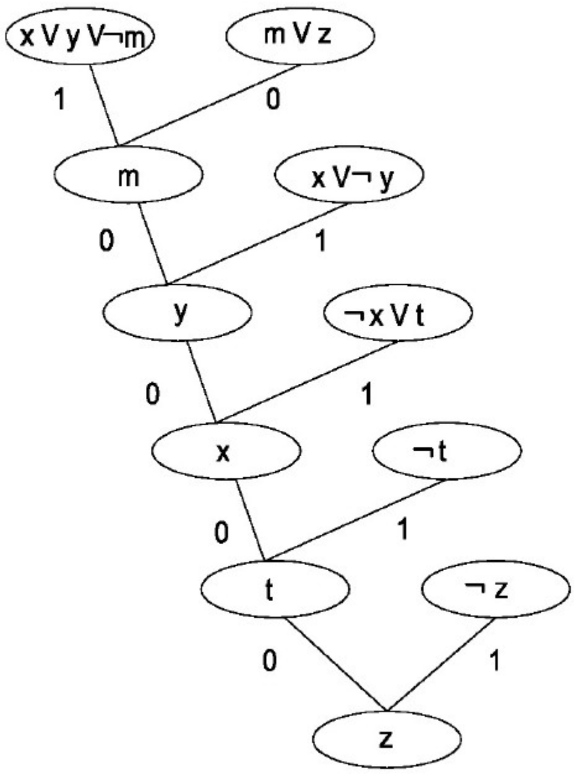
4.2 Booleovský vyhľadávací strom

Regulárny stromový dôkaz π tautológie A môže byť ekvivalentne popísaný aj **booleovským vyhľadávacím stromom**. Je to binárny strom, kde vrcholy sú určené premennými z $\neg A$, okrem listov, ktoré sú označené počiatočnými klauzulami C_i. Z regulárneho rezolučného stromu, ktorým je π reprezentovaný, dokážeme takýto strom zostrojiť tak, že listy ponecháme ako pôvodné klauzuly a postupujeme smerom ku koreňu spôsobom, že hrany doteraz označované vylučovanými literálmi $x, \neg x$ premenujeme na konštantu 0 vtedy, keď prislúchajúci syn (susediaci vrchol ďalej od koreňa) obsahuje literál x, respektíve 1, ak prislúchajúci syn obsahuje literál $\neg x$. Následne vrchol pomenovaný odvodenou klauzulou D_i preznačíme na x. Pri reálnej implementácii si však danú klauzulu musíme ešte chvíľu zapamätať, inak by sme nevedeli ďalej vyznačiť hrany $0, 1$. Po prejdení celého stromu a premenovaní koreňa sme zostrojili booleovský vyhľadávací strom.
Každá cesta v tomto strome korešponduje s čiastočným ohodnotením premenných v $\neg A$, kde premennej x sa priradí hodnota 0 alebo 1 vzhľadom na to, či cesta z vrcholu označenom x pokračuje k synovi po ceste označenej nulou alebo jednotkou.

Pozorovanie 21. *Pre každú cestu v booleovskom vyhľadávacom strome dôkazu π formuly A udávajúcu ohodnotenie α z koreňa do listu označeného klauzulou C_i platí, že α zapríčini nesplnenosť C_i, $C_i(\alpha) = 0$.*

Teda, booleovský vyhľadávací strom rieši vyhľadávací problém, kedy k danému ohodnoteniu α nájde klauzulu z $\neg A$, ktorá je ním nesplnená.

Pozorovanie 22. *Booleovský vyhľadávací strom dôkazu tautológie DNF formule A je zároveň strom reprezentujúci beh $DPLL$ algoritmu na nesplniteľnej CNF formule $\neg A$ nahliadnutý zdola nahor.*



Obr. 4.1: *Booleovský vyhľadávací strom dôkazu formuly* $\neg F$ *z príkladu z predchádzajúcej kapitoly*

5. Princíp holubníku

Pod princípom holubníku PHP_n^m $(m > n)$ rozumieme jednoduchý, no efektívny a elegantný nástroj na dokazovanie rôznych tvrdení prvý raz pomenovaný nemeckým matematikom *J.P.G.L. Dirichletom* ako tzv. **"zásuvkový princíp"**.

PHP_n^m tvrdí, že pokiaľ sa do každej z n priehradok holubníku zmestí najviac jeden holub, vždy existuje aspoň jeden z m holubov, ktorý priehradku nemá.

5.1 Logika v holubníku

Atóm $x_{i,j}$ hovorí o tom, že holub $i \in \{1, ..., m\}$ patrí do priehradky $j \in \{1, ..., n\}$, čím v reči logiky sa formula PHP_n^m skladá z klauzúl $(\bigvee_{j \in \{1,...,n\}} x_{i,j})$ pre každého holuba $i \in \{1, ..., m\}$ a klauzúl $(\neg x_{i_1,j} \vee \neg x_{i_2,j})$ pre všetky dvojice rôznych holubov $i_1, i_2 \in \{1, ..., m\}$ a priehradok $j \in \{1, ..., n\}$. Teda každý holub má svoju priehradku a ani jedna dvojica holubov nemá priehradku spoločnú.

Ďalej sa budeme zaoberať len najsilnejšou variantou, PHP_n^{n+1}. Pokiaľ by $m > (n + 1)$, princíp sa stane slabším, jeho zložitosť klesne, čím bude ľahšie dokázateľným.

Veta 23. *$PHP_n^{n+1} = (\bigwedge_i \bigvee_j x_{i,j}) \wedge (\bigwedge_{i_1 < i_2} \bigwedge_j (\neg x_{i_1,j} \vee \neg x_{i_2,j}))$, $i, i_1, i_2 \in \{1, ..., n+1\}$, $j \in \{1, ..., n\}$, je nesplniteľná CNF formula.*

Dôkaz. Uvažujme ľubovolné ohodnotenie $\alpha : \{x_{i,j} | i, j\} \longrightarrow \{0, 1\}$ a označme $E_\alpha := \{(i, j) | \alpha(x_{i,j}) = 1\}$. Kebyže α spĺňala PHP_n^{n+1}, potom E_α by bol graf prostého zobrazenia z $(n + 1)$-prvkovej množiny do množiny veľkosti n, čo nie je možné. $\square$

Z definície formuly PHP nejde nutne o zobrazenie, ale o multifunkciu, pretože jeden holub môže mať aj dve alebo viac priehradok, no pridaním klauzúl $(\neg x_{i,j_1} \vee \neg x_{i,j_2})$ pre všetky dvojice rôznych priehradok $j_1, j_2 \in \{1, ..., n\}$ a holubov $i \in \{1, ..., m\}$ je možné vynútiť, aby o funkciu išlo.

5.2 Rezolučná hra

Nech F je CNF formula s premennými $x_1, ..., x_n$. Definujme si hru [2], v ktorej dve osoby, **Dokazovateľ (D)** a **Nepriateľ (N)**, konštruujú (čiastočné) ohodnotenie $x_1, ..., x_n$. N tvrdí, že existuje splňujúce ohodnotenie F, D sa mu snaží dokázať opak. V každom kole hry, D zvolí premennú x_i, N vyberie, či jej priradí 0, 1, alebo nechá voľbu na D. V poslednom prípade, ak D nastaví hodnotu x_i sám, N získa bod. Hra skončí v momente, kedy čiastočné ohodnotenie zapríčini nesplnenosť nejakej klauzuly z F, respektíve vtedy, keď všetky premenné budú vyhodnotené. Na nesplniteľnej formule nie je otázkou kto vyhrá, ale koľko bodov je N schopný získať.

Veta 24. *[2] Nech F je nesplniteľná CNF formula. Ak existuje rezolučný strom pre prázdnu klauzulu z F veľkosti najviac S, potom N získa najviac $\log S$ bodov v každej rezolučnej hre hranej na formule F.*

Dôkaz. Buď F nesplniteľná CNF formula s premennými $x_1, ..., x_n$, Π nech je booleovský vyhľadávací strom zostrojený z rezolučného stromu pre prázdnu klauzulu z F. Predpokladajme, že D a N hrajú rezolučnú hru na F, kde úspešne konštruujú ohodnotenie α. Nech α_i je čiastočné ohodnotenie zostrojené po i kolách hry, t.j. α_i priradí i premenným hodnotu 0 alebo 1. Označme p_i počet bodov, ktoré N získal po i kolách a Π_{α_i} podstrom Π, ktorý má koreň v uzle dosiahnutom cestou špecifikovanou α_i.

Najskôr indukciou podľa počtu kôl hry dokážeme pomocné tvrdenie, že pre všetky i platí nerovnosť $|\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}}$. Na začiatku, Π_{α_0} je celý strom a N má 0 získaných bodov, čiže tvrdenie platí. Teraz predpokladajme, že tvrdenie platí aj po i kolách a D vybral premennú x v $(i+1)$. kole. Ak N zvolí jej hodnotu, $p_{i+1} = p_i$ a $|\Pi_{\alpha_{i+1}}| \leq |\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} = \frac{|\Pi|}{2^{p_{i+1}}}$.

Ak N nechal voľbu na D, D použije nasledovnú stratégiu, aby zvolil hodnotu x. Nech $\alpha_i^{x=j}$ je ohodnotenie rozširujúce α_i o $x := j$. D položí $x := 0$ ak $|\Pi_{\alpha_i}^{x=0}| \leq \frac{|\Pi_{\alpha_i}|}{2}$, v opačnom prípade priradí x hodnotu 1. Všimnime si, že pokiaľ D nastaví x na 1, platí $|\Pi_{\alpha_i}^{x=1}| \leq \frac{|\Pi_{\alpha_i}|}{2}$. Čiže ak voľba D je $x := j$, $j \in \{0, 1\}$, dostávame $|\Pi_{\alpha_{i+1}}| = |\Pi_{\alpha_i}^{x=j}| \leq \frac{|\Pi_{\alpha_i}|}{2} \leq \frac{|\Pi|}{2^{p_i+1}} = \frac{|\Pi|}{2^{p_{i+1}}}$. □

5.3 Zložitosť PHP

Teraz môžeme ukázať, že PHP_n^{n+1} je "ťažký"pre stromovú rezolúciu. [2]

Veta 25. *Každý rezolučný strom pre prázdnu klauzulu z PHP_n^{n+1} má veľkosť $2^{\Omega(n)}$.*

Dôkaz. Na dokázanie tejto vety stačí nájsť v rezolučnej hre D a N popísanej vyššie vhodnú stratégiu pre N tak, aby získal aspoň n bodov a aplikovať obmenenú vetu k *Vete 24*.

Priehradka j je obsadená vtedy, keď existuje $i \in \{1, ..., n+1\}$ také, že premennej $x_{i,j}$ bola v hre priradená hodnota 1. Bez ujmy na všeobecnosti predpokladajme, že D sa nepýta na rovnakú premennú viackrát.

N použije nasledovnú stratégiu: Ak sa D spýta na premennú $x_{i,j}$, odpovie 0 pokiaľ je j už obsadená, inak nechá voľbu na D.

Všimnime si, že hra nikdy neskončí nesplniteľnosťou žiadnej z klauzúl $(\neg x_{i_1,j} \vee \neg x_{i_2,j})$. Teda hra nutne musí skončiť na jednej z klauzúl $\bigvee_j x_{i,j}$, t.j. pre nejaké $i \in \{1, ..., n+1\}$ všetkým premenným $x_{i,j}$, $j \in \{1, ..., n\}$ niekto z hráčov priradí hodnotu 0.

Ak D položil $x_{i,j} := 0$, potom N získal jeden bod. Naopak, ak N položil $x_{i,j} := 0$, vzhľadom na stratégiu musel existovať iný holub $i' \neq i$ sediaci v priehradke j, t.j. $x_{i',j} = 1$. Toto rozhodnutie však musel urobiť D, pretože N nikdy nepriradí premennej hodnotu 1. Z toho plynie, že N získa bod za každú premennú $x_{i,j}$, $j \in \{1, ..., n\}$. $\square$

5.4 Rezolučný dôkaz PHP

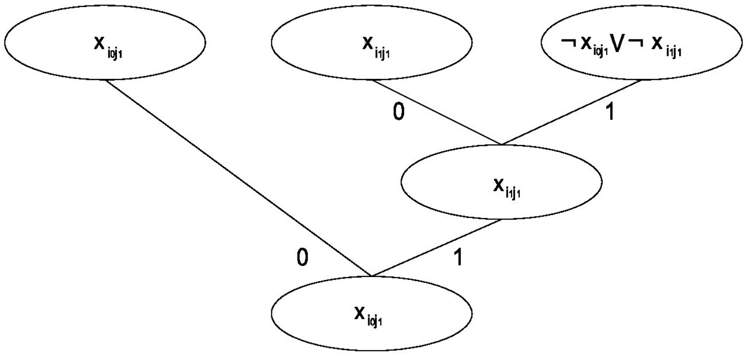
Označme $PHP(I, J)$ formulu $PHP_{|I|}^{|J|}$ zloženú z atómov $x_{i,j}$, $i \in I$, $j \in J$, kde I je množina holubov a J množina priehradok. Uvažujme príklad $PHP(I, J)$, $I = \{i_0, i_1, ..., i_n\}$, $J = \{j_1, ..., j_n\}$.

Veta 26. *[5] Zložitosť stromového rezolučného dôkazu nesplniteľnosti $PHP(I, J)$ je najviac $6(n+1)!$.*

Dôkaz. Indukciou podľa n zostrojme stromový rezolučný dôkaz nesplniteľnosti formuly $PHP(I, J)$.

Predpokladajme, že D a N hrajú rezolučnú hru na $PHP_{|I|}^{|J|}$, pričom navrhneme vhodnú stratégiu pre D na voľbu premenných.

$I = \{i_0, i_1\}$, $J = \{j_1\}$, $PHP(I, J) = (x_{i_0,j_1}) \wedge (x_{i_1,j_1}) \wedge (\neg x_{i_0,j_1} \vee \neg x_{i_1,j_1})$. Nech D zvolí ľubovolnú premennú formuly (je jedno ktorú, sú symetrické). Pokiaľ N nechce, aby hra skončila hneď, musí jej priradiť hodnotu 1. Avšak v druhom kole, pri všetkých možných hodnotách zostávajúcej premennej, ohodnotenie zapríčini nesplnenosť $PHP(I, J)$ a teda vieme skonštruovať binárny vyhľadávací strom zodpovedajúci rezolučnému stromu pre prázdnu klauzulu z $PHP(I, J)$, viď. *Obr. 5.1.*



Obr. 5.1: *Binárny vyhľadávací strom zodpovedajúci rezolučnému stromu pre prázdnu klauzulu z* $PHP(I, J)$ *pre* $n = 1$

Prípad $n = 1$ je teda zrejmý. Pozrime sa teraz na indukčný krok $(n-1) \longrightarrow n$. Predpokladajme, že pre holuba $i_0 \in I$ a ľubovolnú priehradku $j_u \in J$ máme stromový rezolučný dôkaz nesplniteľnosti $PHP(I \setminus \{i_0\}, J \setminus \{j_u\})$.

D sa postupne bude pýtať na atómy $x_{i_0,j_1}, x_{i_0,j_2}, ..., x_{i_0,j_n}$. Ak každá odpoveď N bude 0, klauzula $(\bigvee_k x_{i_0,j_k})$ sa stane nesplniteľnou. Nech teda x_{i_0,j_l} je prvá nenulová odpoveď N.

D ďalej zafixuje j_l a pýta sa na ostatné atómy množiny I, t.j. postupne $x_{i_1,j_l}, ...,$ x_{i_n,j_l}. Ak na tieto atómy čo i len jedna odpoveď N bude 1 (x_{i_v,j_l}), opäť dostaneme nesplniteľnosť pôvodnej klauzuly $(\neg x_{i_0,j_l} \vee \neg x_{i_v,j_l})$. Pokiaľ sú všetky odpovede 0, bezpečne môžeme odstrániť prvky $i_0 \in I, j_l \in J$.

Keďže z indukčného predpokladu máme binárny vyhľadávací strom zodpovedajúci rezolučnému odvodeniu prázdnej klauzuly z $PHP(I \setminus \{i_0\}, J \setminus \{j\})$, dostávame obdobný binárny vyhľadávací strom aj pre $PHP(I, J)$, viď. *Obr. 5.2.*

Takýto binárny vyhľadávací strom má veľkosť

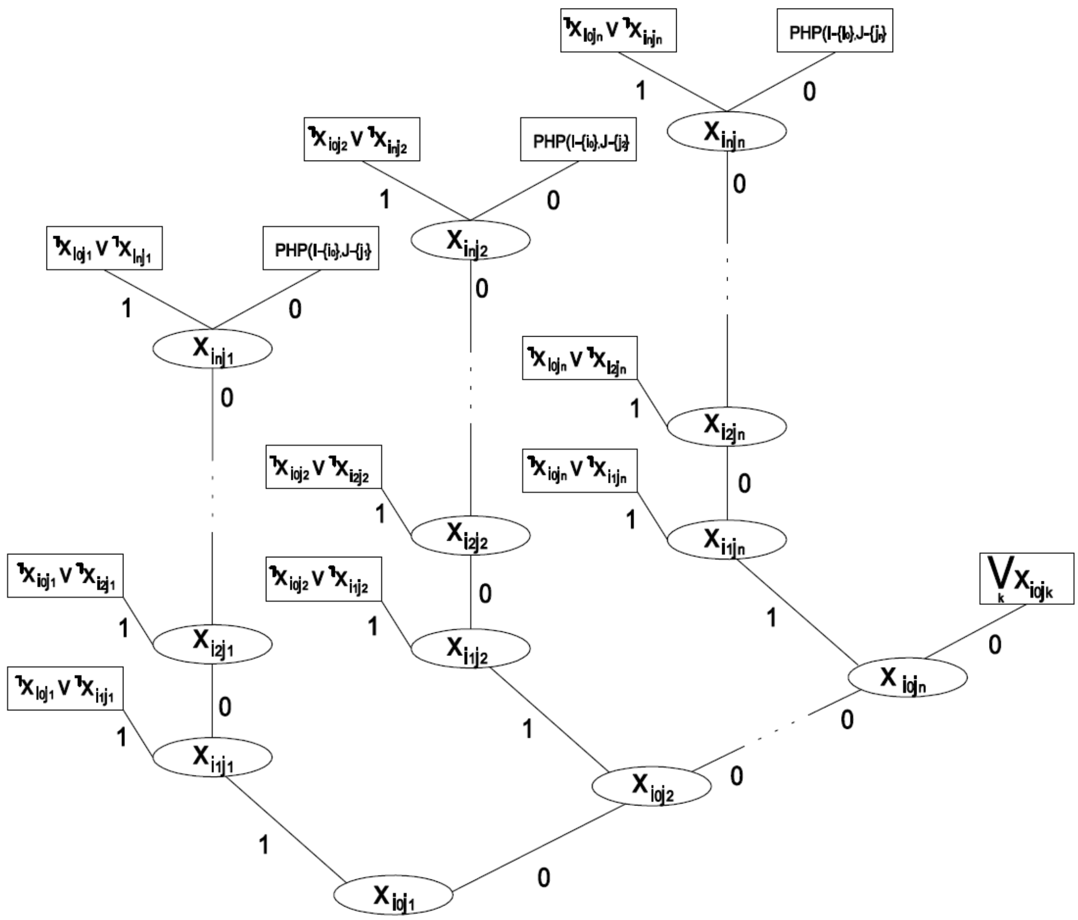
$$S(n) = \begin{cases} nS(n-1) + n(2n+1) + 1 & \text{ak } n > 1 \\ 5 & \text{ak } n = 1, \end{cases}$$

kde $S(n-1)$ je veľkosť stromu pre PHP^n_{n-1}.

Teraz už je jednoduché indukciou ukázať, že $S(n) \leq 6(n+1)!$. □

Zostrojený strom rezolučného dôkazu nám ukazuje jednoduchú, no efektívnu heuristiku pre $DPLL$ algoritmus na formule PHP^{n+1}_n. Pokiaľ atóm ohodnotíme jednotkou, množstvo ďalších môžeme automaticky ohodnotiť nulou, pretože už žiaden iný holub sa do obsadenej priehradky nezmestí, no naopak žiadnu inú pridanú informáciu nedostaneme. Teda každé čiastočné ohodnotenie je určené nielen jednotkami priradenými jednotlivým premenným, ale aj všetkými nulami, ktoré z priradení vyplynú.

Pri *DPLL* algoritme bez použitej heuristiky by korešpondujúci výrokový dôkaz nesplniteľnosti PHP_n^{n+1} mal veľkosť $(2^n - 1)^{n+1}$, čiže by bol zložitostne ekvivalentný tabuľke pravdivostných hodnôt.



Obr. 5.2: *Binárny vyhľadávací strom zodpovedajúci rezolučnému odvodeniu prázdnej klauzuly z $PHP(I, J)$*

Záver

V práci bol na jednej strane predstavený základný algoritmus na rozhodovanie o splniteľnosti výrokových formúl, na strane druhej výrokové dôkazové systémy na princípe rezolúcie, všetko podložené všeobecne známou teóriou z oblasti logiky a zložitosti.

Finálne sa vcelku jednoducho ukázalo nie len, že beh DPLL algoritmu na nesplniteľnej formule môže byť chápaný ako výrokový dôkaz jej nesplniteľnosti, ale aj naopak, na daný rezolučný strom pre prázdnu klauzulu sa vieme pozrieť ako na beh algoritmu.

V piatej kapitole boli na príklade princípu holubníku ilustrované praktické dôsledky tohto tvrdenia, nakoľko nám zostrojený rezolučný dôkaz ukázal heuristiku pre algoritmus, ktorá mu ušetrila nezanedbateľné množstvo času.

Zoznam použitej literatúry

[1] BEAME, Paul. KARP, Richard. PITASSI, Toniann. SAKS, Michael. The Efficiency of Resolution and Davis-Putnam Procedures. *SIAM Journal on Computing*. Volume 31 Issue 4, 2002. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

[2] BEYERSDORFF, Olaf. *Proof and Games*. Institut für Theoretische Informatik, Leibniz-Universitat Hannover, Germany. Course Material ESSLLI 2011 Ljubljana.

[3] COOK, Stephen. The complexity of theorem proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. 151–158. 1971.

[4] COOK, Stephen. RECKHOW, Robert. The Relative Efficiency of Propositional Proof Systems. *The Journal of Symbolic Logic*. Volume 44, Issue 1, 1979. pp. 36-50.

[5] DANTCHEV, S. RIIS, S. *Tree resolution proofs of the weak pigeon-hole principle*. 16th Annual IEEE Conference on Computational Complexity, 18-21 June 2001, Chicago, Illinois. New York: IEEE, 2001, pp. 69-77.

[6] HOFFMANN, Jan. *Resolution Proofs and DLL Algorithms with Clause Learning*. München, 25. September 2007. Diploma Thesis. Institut für Informatik der Ludwig-Maximilians-Universität München.

[7] IMPAGLIAZZO, Russell. PUDLÁK, Pavel. *A lower bound for DLL algorithms for k-SAT*. July 12, 1999.

[8] KOSTOLÁNYI, Peter. *Dôkazová zložitosť*. Proseminár z informatiky, 5.3. 2012. Fakulta matematiky, fyziky a informatiky, Univerzita Komenského, Bratislava.

[9] KRAJÍČEK, Jan. *Propositional proof complexity I*. Praha: Matematical institute, Academy of Sciences of the Czech Republic, 2003.

[10] KRAJÍČEK, Jan. PUDLÁK, Pavel. Propositional proof systems, the consistency of first order theories and the complexity of computations. *JSL*. Vol.54, No.3, 1989. pp. 1063-1079.

[11] LOMITZKI, Jan. *Řešení problému splnitelnosti booleovské formule (SAT)*. Praha, 18.1. 2008. Bakalářská práce. České vysoké učení technické, Fakulta elektrotechnická.

[12] Švejdar, Vítězslav. *Logika: neúplnost, složitost a nutnost.* Academia Praha, 2002. ISBN 80-200-1005-X.

Zoznam obrázkov