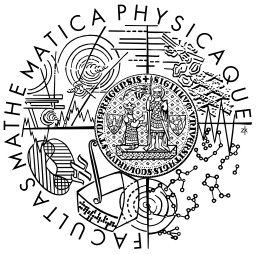

MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Václav Luňák

STP řešič pro OpenSMT

Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: doc. RNDr. Jan Kofroň, Ph.D.

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Poděkování.

Název práce: STP řešič pro OpenSMT

Autor: Václav Luňák

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: doc. RNDr. Jan Kofroň, Ph.D., Katedra distribuovaných a spolehlivých systémů

Abstrakt: Abstrakt.

Klíčová slova: klíčová slova

Title: STP solver for OpenSMT

Author: Václav Luňák

Department: Department of distributed and dependable systems

Supervisor: doc. RNDr. Jan Kofroň, Ph.D., Department of distributed and dependable systems

Abstract: Abstract.

Keywords: key words

Obsah

Úvod

1. Analýza

1.1 Fungování SMT řešičů

Problém splnitelnosti booleovské formule, označovaný též zkratkou SAT, patří k nejznámějším problémům z oboru matematické logiky. V roce 1971 se stal prvním dokazatelně NP-úplným problémem [2] a v důsledku toho i užitečným nástrojem pro teorii složitosti, pomocí něhož lze rozhodovat o NP-úplnosti dalších problémů.[1]

V praktickém použití však SAT naráží na své limitované vyjadřovací schopnosti. Práce s binárními proměnnými, omezenými pouze na dvě různé hodnoty, může komplikovat převod reálného problému do tvaru booleovské formule. Potřeba užití komplexnějších atomů tak vedla ke zobecnění SAT zvanému *Satisfiability modulo theories* (SMT).

Jak název napovídá, SMT rozšiřuje SAT o jazyk logických teorií. Máme-li nějakou teorii T, pak instancí SMT rozumíme formuli jazyka této teorie. Jiným pohledem můžeme na instanci SMT nahlížet jako na booleovskou formuli, ve které jsme nahradili některé binární proměnné za predikáty obsažené v T. Problému vztaženému k této konkrétní teorii pak říkáme *SMT s ohledem na T*.

Jako příklady teorií tradičně řešených v SMT lze uvézt např. teorii lineární aritmetiky (LA), teorii neinterpretovaných funkcí s rovností (EUF), či teorii diferenční logiky (DL), kterou se budeme zabývat ve zbytku práce.

Vzhledem k podobnostem mezi SAT a SMT není překvapením, že SMT řešiče využívají schopností SAT řešičů. Přechod mezi rozhraním termů teorie a kapacitami SAT řešiče zpravidla probíhá jedním ze dvou základních způsobů [5].

První přístup se nazývá *hladový*. Hladové SMT řešiče operují ve dvou krocích. V prvním převedou celou vstupní formuli na ekvisplnitelnou booleovskou formuli. Druhý krok pak již spočívá jen v předání této formule existujícímu SAT řešiči. Pokud bychom tedy pracovali například s aritmetikou nad osmibitovými čísly, mohli bychom reprezentovat každou proměnnou osmi binárními proměnnými a aritmetické operace převézt na odpovídající sekvence logických operací.

Nespornou výhodou hladových řešičů je možnost použití již existujících metod verifikace implementovaných na řešení SAT. Pro některé teorie se také hladové řešiče ukazují být rychlejší než jejich alternativy. Jejich největší problém pak obecně spočívá v překladu literálů teorie do booleovských formulí. Ten musí být zkonstruován samostatně pro každou teorii, a navíc může v závislosti na teorii produkovat formule znatelně delší, než byl původní vstup. Efektivní převody existují např. pro EUF s omezenou doménou [10], obecně však hladový přístup není příliš rozšířený.

Jeho protějškem je takzvaný *líný přístup*. Líný přístup se nesnaží měnit strukturu vstupní formule; namísto toho je každý predikát abstrahován pomocí nové binární proměnné. Když pak SAT řešič rozhodne o ohodnocení těchto proměnných, oznámí toto rozhodnutí *theory solveru* pro danou teorii. Theory solver je schopen určit, zda je dané ohodnocení správné, tzn. dokáže rozhodnout o splnitelnosti nějaké konjunkce literálů teorie. SAT řešič potom hledá platná ohodnocení, dokud nenajde takové, které theory solver prohlásí za konzistentní.

Ve prospěch líných SMT řešičů svědší fakt, že pro danou teorii často existují dobře známé postupy na ověření konjunkce literálů. Pro LA například můžeme využít metod

[1] Příklady převodů NP-úplných problémů na SAT můžeme nalézt např. v [8, kapitola 19]

lineárního programování, theory solver pro DL řeší STP (viz. 1.2) a podobně. Mohou však ztrácet efektivitu zejména v důsledku *slepého prohledávání* [9], kde hlavní řešič rozhoduje o hodnotě predikátů, aniž by a priori věděl o důsledcích těchto ohodnocení v rámci teorie, což může vést k nutnosti vyzkoušet velké množství ohodnocení, než je nalezeno nějaké, které je s teorií konzistentní.

V roce 2004 navrhli Gazinger a kol. nový přístup zvaný *DPLL(T)* [7]. DPLL(T) má koncepčně blíže k línému vyhodnocování, integruje však těsněji hlavní řešič s theory solverem. Místo toho, aby využíval theory solveru až po nalezení nějakého ohodnocení, průběžně mu oznamuje dosavadní rozhodnutí a periodicky se ho ptá na splnitelnost právě dosazené konjunkce. Theory solver pak kromě kontroly splnitelnosti také oznamuje hlavnímu řešiči důsledky této konjunkce. Tím jsme schopni dříve opustit větve rozhodovacího stromu nekonzistentní s teorií.

V jádru tohoto přístupu stojí všeobecný DPLL(X) engine, využívající DPLL [3] postupu pro SAT řešiče. Tento engine nemusí mít žádné znalosti o konkrétní teorii. Dosazením theory solveru $Solver_T$ pro danou teorii T za parametr X pak vytvoříme konkrétní instanci DPLL(T). Hlavní engine komunikuje se $Solver_T$ pomocí následujícího rozhranní [7]:

Initialize(L: množina literálů). Inicializuje $Solver_T$ s L jakožto množinou literálů, které se vyskytují v problému.

SetTrue(l: L-literál): množina L-literálů. Skončí výjimkou, pokud se l ukáže jako nekonzistentní s dosud zadanými literály teorie. V opačném případě přidá l do seznamu zadaných literálů a vrátí množinu L-literálů, které jsou důsledky přidání l do tohoto seznamu.

IsTrue?(l: L-literál): boolean. Vrátí *true* právě tehdy, když l je důsledkem seznamu přidaných literálů. *false* tedy vrací, pokud je důsledkem tohoto seznamu $\neg l$, nebo pokud z něj nevyplývá ani l, ani $\neg l$.

Backtrack(n: přir. číslo). Odstraní posledních n hodnot ze seznamu zadaných literálů. n nesmí být větší než velikost tohoto seznamu.

Explain(l: L-literál): množina L-literálů. Vrátí pokud možno co nejmenší podmnožinu zadaných literálů, z jejichž konjunkce plyne l. Pro l musí platit, že je důsledkem nějaké takové podmnožiny, tedy musí být obsažen v návratové hodnotě nějakého volání **SetTrue(l')** takového, že l' nebylo zahozeno žádným následným voláním **Backtrack**.

Při použití tohoto rozhranní přitom $Solver_T$ nemusí nic vědět o implementaci DPLL(X) enginu. Framework je tedy velice modulární a snadno rozšiřitelný o nové teorie. Vyžadujeme pouze, aby byl $Solver_T$ schopný inkrementálně přijímat a odebírat jednotlivé predikáty teorie. Tento postup se v praxi ukazuje jako efektivnější než dostupné alternativy. Většina dnes rozšířených SMT řešičů — včetně námi používaného OpenSMT2 — je tedy založena na metodě DPLL(T).

1.2 Rozbor STP

Jedním ze základních podproblémů vyskytujícím se v takřka všech plánovacích problémech je takzvaný Simple Temporal Problem (STP). STP poprvé postulovali v roce

1991 Dechter, Meiri a Pearl [4] a od té doby našel široká využití jak v informatických oblastech, tak v oborech od medicíny [1] po vesmírný let [6].

Vstupem STP je množina rozdílových omezení, to jest nerovnic tvaru

$$x - y \leq c,$$

kde x a y jsou proměnné a c je konstanta. V závislosti na tom, jakou verzi problému řešíme, přitom pracujeme buď s celočíselnými, nebo s reálnými hodnotami. Výstupem tohoto problému je pak rozhodnutí, zda existuje ohodnocení proměnných tak, aby byla splněna všechna zadaná omezení. V rozšíření problému pak můžeme požadovat na výstupu i nějaké takovéto splnitelné ohodnocení, pokud existuje, případně nalezení pokud možno co nejmenší podmnožiny omezení, která zajišťuje nesplnitelnost problému.

Na první pohled se může zdát pevně daný tvar nerovnic příliš omezující, uvědomme si však, že do této formy můžeme převést několik dalších druhů nerovnic. Nejsnáze zahrneme do problému omezení tvaru $x - y = c$; ty stačí jednoduše nahradit nerovnicemi $x - y \leq c$ a $x - y \geq c$.

Problematické nejsou ani nerovnice typu $\pm x \leq c$. Pro účely takovýchto omezení si zavedeme novou globální proměnnou $zero$, s jejíž pomocí převedeme předchozí do tvaru $x - zero \leq c$, respektive $zero - x \leq c$. Pokud pak hledáme splňující ohodnocení proměnných, najdeme takové, kde $zero$ je ohodnoceno nulou. Korektnost tohoto postupu zaručuje následující tvrzení.

Lemma 1. *Je-li σ splňující ohodnocení nějakého STP a ε libovolná konstanta, pak ohodnocení π definované pro všechny proměnné x jako $\pi(x) = \sigma(x) + \varepsilon$ je také splňující ohodnocení tohoto STP.*

Důkaz. Plyne okamžitě z tvaru rozdílových omezení. □

Můžeme do problému zahrnout taktéž omezení tvaru $x - y < c$. Pro celočíselné proměnné lze tuto nerovnici ekvivalentně zapsat jako $x - y \leq c - 1$. V reálné variantě pak nahradíme nerovnici výrazem $x - y \leq c - \delta$, přičemž nenastavujeme okamžitě konkrétní hodnotu δ, ale udržujeme si ji pouze symbolicky a určujeme její vhodné dosazení až při výpočtu splňujícího ohodnocení. Tento postup je detailněji popsán v sekci 2.5. Uvědomme si, že pokud jsme schopni vyjádřit ostré nerovnosti, umíme vyjádřit i negace neostrých nerovností a naopak.

V jazyce výrokové logiky pak teorii obsahující výše popsané nerovnice nazveme *teorie diferenční logiky* a budeme ji značit *DL*. Celočíselnou variantu této teorie pak budeme značit jako *IDL* a reálnou variantu jako *RDL*. Nahradíme-li pak v booleovské formuli některé termy těmito nerovnicemi, ověření splnitelnosti takto vzniklé formule je instancí SMT problému s ohledem na DL.

1.3 Převod na grafový problém

Velkou rozšířenost STP můžeme mimo jiné přisoudit tomu, že jsme schopni ho efektivně řešit. Jelikož se problém skládá výlučně z lineárních omezení, mohli bychom na první pohled využít metod lineárního programování, jako je například simplexový algoritmus. Tyto metody jsou schopny řešit i mnohem komplexnější problémy, avšak s jejich výpočetní silou se pojí znatelně vyšší časová náročnost. Algoritmy specializované na STP se proto už od svého počátku [4, Kapitola 2] obracejí jiným směrem,

a to k formalizmu teorie grafů. Přestože v průběhu let vzikly různé metody řešení tohoto problému, všechny fungují na základě převedení množiny omezení na takzvaný *omezující graf*.

Definice 1 (Omezující graf)**.** *Nechť* Π *je množina rozdílových omezení. Omezujícím grafem této množiny rozumíme hranově ohodnocený orientovaný graf G takový, že vrcholy G tvoří proměnné vyskytující se v* Π *a každému omezení* $(x - y \leq c) \in \Pi$ *odpovídá v G hrana* $\langle x,y \rangle$ *s ohodnocením c.*

Poznámka. Hranu $\langle x,y \rangle$ s ohodnocením c budeme značit $x \xrightarrow{c} y$. Orientovanou cestu z x do y se součtem ohodnocení k pak budeme značit $x \xrightarrow{k*} y$.

Pro úplnost dodejme, že dvojice proměnných se může vyskytovat v libovolně mnoha omezeních. Omezující graf je tedy formálně orientovaným multigrafem. Vzhledem k vzájemné bijekci mezi hranami grafu a nerovnicemi problému budeme v průběhu práce volně přecházet mezi oběma reprezentacemi.

Převod do formy grafu je pro řešení problému zásadní. Umožňuje nám totiž formulovat následující klíčové tvrzení.

Tvrzení 2. *Nechť* Π *je množina rozdílových omezení. Instance STP tvořená touto množinou je splnitelná právě tehdy, když omezující graf* Π *neobsahuje záporné cykly.*

Důkaz. Najdeme-li v omezujícím grafu záporný cyklus obsahující vrchol x, sečtením všech nerovnic vyskytujících se v tomto cyklu dostaneme $x - x \leq c < 0$, z čehož je jasně problém nesplnitelný. Je-li na druhou stranu problém nesplnitelný, obsahuje Π nějakou nerovnici $x - y \leq c$ takovou, že z Π vyplývá $y - x < -c$. Tato implikace znamená, že v omezujícím grafu existuje cesta $y \xrightarrow{k*} x$ taková, že $k < -c$. Hrana $x \xrightarrow{c} y$ pak společně s touto cestou tvoří záporný cyklus. □

Hledání splnitelnosti STP jsme tedy schopni převést na hledání záporného cyklu v grafu. To je problém, který dokážeme efektivně řešit. Využít můžeme např. některý algoritmus na hledání nejkratší cesty, kupříkladu Floydův-Warshallův algoritmus operující v čase $\Theta(|V|^3)$ nebo Bellmanův-Fordův algoritmus, který má časovou složitost $\Theta(|V| \cdot |E|)$.

Tyto algoritmy však trpí pro náš účel zásadním nedostatkem. Jejich použití znamená, že po každém přidání nové hrany do grafu musí znovu proběhnout celé prohledávání. Tento postup není vhodný pro použití v SMT řešičích, ve kterých je kladen velký důraz na inkrementalitu. V následující sekci tedy podrobně rozebereme několik postupů pro řešení problémů SMT s ohledem na DL a motivujeme výběr námi použitého algoritmu.

1.4 Volba algoritmu

2. Popis řešení

2.1 Prostředí

2.2 Úpravy referenčního algoritmu

2.3 Datové struktury

2.4 Popis běhu programu

2.5 Srovnání reálné a celočíselné verze

3. Programátorská dokumentace

3.1 Přidávání literálů

3.2 Hledání důsledků

3.3 Rozhodování o splnitelnosti

3.4 Hledání konfliktů a backtracking

3.5 Nalezení splňujícího ohodnocení

4. Experimentální měření

4.1 Metodologie

4.2 Výsledky

4.3 Srovnání

Závěr

Seznam použité literatury

[1] Anselma, L., Terenziani, P., Montani, S. a Bottrighi, A. Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines. *Artificial Intelligence In Medicine*, **38**(2):171 – 195, 2006.

[2] Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[3] Davis, M. a Putnam, H. A computing procedure for quantification theory. *J. ACM*, **7**(3):201–215, July 1960.

[4] Dechter, R., Meiri, I. a Pearl, J. Temporal constraint networks. *Artificial Intelligence*, **49**(1-3):61–95, 1991.

[5] Etessami, K., Rajamani, S. K., Nieuwenhuis, R. a Oliveras, A. *DPLL(T) with Exhaustive Theory Propagation and Its Application to Difference Logic.*, page 321. 2005.

[6] Fukunaga, A., Rabideau, G., Chien, S. a Yan, D. Towards an application framework for automated planning and scheduling. *1997 IEEE Aerospace Conference, Aerospace Conference, 1997. Proceedings., IEEE*, **1**:375, 1997.

[7] Harald, G., George, H., Robert, N., Albert, O. a Cesare, T. Dpll(t): Fast decision procedures. *Computer Aided Verification : 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings*, page 175, 2004.

[8] Mareš, M. a Valla, T. *Průvodce labyrintem algoritmů*. CZ.NIC, 2017.

[9] Moura, L. a Ruess, H. An experimental evaluation of ground decision procedures. *Computer Aided Verification : 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings*, page 162, 2004.

[10] Randal E., B., Shuvendu K., L. a Sanjit A., S. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions. *Computer Aided Verification : 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings*, page 78, 2002.

A. Přílohy

A.1 První příloha