Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Implementace důkazového systému pro výrokovou logiku

Jan Švajcr

Vedoucí práce: Mgr. Jan Starý, Ph.D.

5. května 2014

Poděkování

Děkuji svému vedoucímu práce Mgr. Janu Starému, Ph.D. za přívětivost a zájem, své rodině za podporu a zázemí a své milované za lásku a věrnost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 5. května 2014 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Švajcr, Jan. *Implementace důkazového systému pro výrokovou logiku.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstrakt

Cílem této práce je vypracovat konzolový program implementující prostředí důkazového systému výrokové logiky. Jeho základní funkcionalitou je syntaktická analýza textového vstupu v podobě posloupnosti výrokových formulí a ověřování, zdali je tato posloupnost korektním výrokovým důkazem. Software je podporován prostředím UNIX a implementován v jazyce C. Součástí práce jsou náležitosti jako dokumentace zdrojového kódu a vytvoření uživatelské příručky v podobě standardní manuálové stránky.

Klíčová slova Výroková logika, Dokazatelnost, Syntaktická analýza, C/C++.

Abstract

The goal of this thesis is a creation of a console program implementing the environment of the proof system of propositional logic. The basic functionality contains parsing of the input represented as a sequence of propositional formulas and validation of this sequence as a proof. The software is implemented in the C language and supported in UNIX systems. This thesis also requires a code documentation and a standard manual page as a user manual.

Keywords Propositional logic, Proofs, Parsing, C/C++.

Obsah

Seznam obrázků

Úvod

Logika je formální věda zkoumající část lidského myšlení. Jejím předmětem je správné vyvozování důsledků z předpokladů, jejichž volbu, pravdivost nebo snad smysl blíže nezkoumáme. Činíme tak nejen proto, že naše vyvození je správné i v případě, kdy předpoklady nejsou, ale i proto, že to této disciplíně ani nepřísluší. Matematická logika toto usuzování formalizuje, čímž nás oprošťuje od psychologického náhledu. Dává tak vzniku postupům, které lze kdykoliv opakovaně aplikovat. Příkladem takového postupu je ověřování korektnosti našeho usuzování, takzvaného *důkazu*. To je dokonce natolik mechanické, že jej můžeme svěřit strojovému zpracování[1]. Právě tento aspekt výrokové logiky byl podnětem pro vznik této práce, která formalismus výrokové logiky přenáší do oblasti informačních technologií implementací některých principů výrokového počtu počítačovým programem. Již z názvu této práce vyplývá, že implementuje principy důkazového systému, konkrétně Hilbertova systému, na který se v této práci omezíme. Dokazatelnost je tedy ústřední kapitola výrokového počtu, o kterou se budeme zajímat.

Kapitola 1

Formální kontext

V úvodní kapitole se seznámíme s několika potřebnými základními pojmy výrokového počtu, aby každý čtenář byl srozumen s terminologii užitou v tomto textu a porozuměl tak obsahu následujících kapitol.

1.1 Výrok

Kvůli zavedení ústředního pojmu *formule* nejprve zavedeme elementární pojem *výrok*. Výrok je takové tvrzení, o kterém má dostatečný smysl uvažovat, zdali je pravdivé či nikoliv. Lze ho tedy také chápat jako oznamovací větu. Výrokům přiřazujeme pravdivostní hodnotu *true* nebo *false*. Toto pak nazýváme *pravdivostním ohodnocením* výroků. Ze základních výroků tvoříme další výroky (tzv. *složené výroky*), pomocí logických operací, které si brzy popíšeme. V závislosti na pravdivostním ohodnocení elementárních výroků a na typu použité logické operace přisuzujeme pravdivostní ohodnocení výrokům z těchto výroků složených. Jednotlivé základní výroky značíme velkými písmeny latinky: $A, B, C, ...$

Pro ilustraci elementárního výroku uvažujme následující výrok A: „Dnes je hezký den.“. Je na čtenáři, aby rozhodl o pravdivosti tohoto výroku a jistě bude souhlasit, že jiný čtenář by mohl rozhodnout stejně nebo opačně. V této práci se ale pravdivostním ohodnocením výroků stejně jako jejich významem nebudeme zabývat. Zmíněný fakt slouží pouze jako součást definice a čtenář ji nemusí považovat za klíčovou.

1.1.1 Logické operace

Logické operace v logice dělíme na unární a binární v závislosti na jejich aritě[1]. Reprezentují je příslušné logické spojky následovně:

Unární
- negace ($\neg$)

[1] Arita - počet operandů operace potřebných k jejímu provedení.

Binární
- konjunkce ($\wedge$)
- disjunkce ($\vee$)
- implikace ($\rightarrow$)
- ekvivalence ($\leftrightarrow$)

Již víme, že logické operace dávají vzniku složeným výrokům, jejichž pravdivostní hodnota závisí na typu použité operace1.1. Význam těchto operací nebudeme blíže sledovat, protože nejsou podstatné pro účely této práce. Pro nás důležitou vnímejme pouze rozlišení jednotlivých operací a jejich aritu.

Pro ilustraci složeného výroku uvažujme následující výrok $B = \neg A$. V souvislosti s předchozím příkladem elementárního výroku prohlásíme, že se jedná o výrok s opačnou pravdivostní hodnotou než má výrok A. Do přirozeného jazyka bychom jej tedy přeložili jako „Dnes není hezký den“ nebo „Není pravda, že dnes je hezký den.“. Druhá varianta překladu více koresponduje se syntaktickým vnímáním tohoto složeného výroku.

1.2 Formule

Ústředním pojmem pro tuto práci je *formule*. V matematické logice obecně definujeme formuli tak, že:

1. Každý elementární výrok je formulí.
2. Vznikne-li α unární logickou operací z formule β nebo binární logickou operací z formulí β a γ , je α také formulí.
3. Každou formuli dostaneme postupnou aplikací předchozích pravidel[1].

Jednotlivé formule značíme malými písmeny řecké abecedy: $\alpha, \beta, \gamma, ...$

Pro ilustraci uvažujme následující formule α a β:

1. $\alpha = A$
2. $\beta = \neg(A \vee B)$

Je zřejmé, že formule α je ve skutečnosti elementárním výrokem, kdežto formule β je složená z několika výroků. Pro názornost popíšeme vznik formule β na základě předchozí definice formule.

1. Uvažujme elementární výroky A a B.
2. Výroky A a B pojí operace disjunkce do podoby složeného výroku $A \vee B$.
3. Na dosavadní formuli aplikujeme unární operaci negace. Tímto vzniká nová formule $\beta = \neg(A \vee B)$.

1.2.1 Syntaxe

Nyní se na okamžik věnujme způsobu zápisu výrokových formulí. Tuto znalost budeme později potřebovat při analýze. Výrokové formule stejně jako složené výroky lze zapisovat ve třech různých notacích:*prefixní*, *infixní* a *postfixní*. Tyto notace jsou navzájem ekvivalentní, liší se pouze pořadím zápisu logické operace u složených výroků. Prefixní, infixní a postfixní zápis názorně předvedeme následovně na jedné formuli:

prefix $\vee AB$

infix $A \vee B$ nebo striktně $(A \vee B)$

postfix $AB\vee$

Je třeba brát na vědomí, že s rostoucí složitostí výrokové formule je pro nás čím dál obtížnější udržet pozornost nad strukturou formule. Všimněme si, že infixní zápis je našemu vnímání nejpřirozenější. Budeme jej proto nadále používat stejně jako doposud.

1.3 Důkazový systém

Důkazový systém je aparát výrokové logiky, který rozhoduje o dokazatelnosti libovolné výrokové formule φ v kontextu daného axiomatického systému. Ten je tvořen dvěma důležitými množinami: množinou axiomů a množinou odvozovacích pravidel, která definují způsob, jakým lze z výrokových formulí odvozovat formule další.

Axiom je speciální výroková formule s abstraktními elementárními členy, za které lze dosazovat konkrétní výrokové formule. Axiom lze tedy chápat jako jakousi šablonu, na základě které lze vytvářet jeho instance. Instancí axiomů je nekonečně mnoho stejně jako výrokových formulí. O pravdivosti axiomů rozhodujeme vždy tak, že je považujeme automaticky za pravdivé.

1.3.1 Hilbertův systém

Jedním takovým formálním důkazovým systémem je Hilbertův axiomatický systém. Ten se omezuje pouze na dvě logické spojky: negaci a implikaci. Tyto totiž dokáží zastoupit veškeré ostatní spojky, které se dají kombinací negace a implikace adekvátně vyjádřit. Tato práce bude implementovat právě tento axiomatický systém.

Množina axiomů obsahuje tyto prvky:

A1: $\varphi \to (\psi \to \varphi)$

A2: $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$

A3: $(\neg\varphi \to \neg\psi) \to ((\neg\varphi \to \psi) \to \varphi)$

Množina odvozovacích pravidel obsahuje jedinný prvek - tzv. pravidlo *modus ponens*, které zavedeme následovně: φ lze odvodit, platí-li ψ zároveň s $\psi \rightarrow \varphi$.

1.3.2 Důkaz

Buď φ výroková formule. Řekněme, že konečná posloupnost výrokových formulí $\varphi, ..., \varphi$ je *důkazem* formule φ ve výrokové logice, pokud φ_n je formule φ, a každá formule φ_i z této posloupnosti je buďto instancí některého axiomu, nebo je z některých předchozích φ_j, φ_k, kde $j, k < i$, odvozena příslušným odvozovacím pravidlem důkazového systému. Libovolná formule φ je dokazatelná (píšeme $\vdash \varphi$), právě když existuje její důkaz. Je zřejmé, že každý důkaz musí začínat axiomem, tedy triviálně každá instance axiomu je dokazatelnou formulí [2].

1.3.2.1 Důkaz z předpokladů

V kontextu výrokového důkazu často také mluvíme o tzv. *předpokladech*. Jedná se o formule, ze kterých v rámci důkazu vycházíme a nedokazujeme je. Mluvíme potom o více obecném důkazu, tzv. *důkazu z předpokladů*. Tento typ důkazu tato práce neimplementuje. Zmínka o něm je zde pouze pro účely pozdější diskuze nad rozšířitelností této práce.

1.3.2.2 Zjednodušování důkazu

V rámci pozdějšího vymezení požadavků na implementovaný software je nutné si ujasnit, jakým způsobem lze důkazy optimalizovat v Hilbertově axiomatickém systému.

Lemma: Buď $\phi_1, \ldots, \phi_n$ posloupnost formulí tvořící Hilbertovský důkaz. Pak kaľdá podposloupnost, která vznikne z $\phi_1, \ldots, \phi_n$ vynecháním druhého a kaľdého daląího výskytu nějaké zvolené formule ϕ_i, je opět důkazem. Tedy i po vynechání vąech duplicit zůstává daná posloupnost důkazem.

Důkaz: Netriviální je jen případ pouľití modus ponens. Pokud ale nějaká formule ϕ_i vyplývá z nějakých předchozích formulí ϕ_j a $\phi_j \rightarrow \phi_i$ pomocí modus ponens, pak stejným způsobem vyplývá i z kaľdých jejich předchozích (specielně z prvních) výskytů.

Tento obrat můľeme povaľovat za zárodek *optimalizace důkazů*, která vąak dalece přesahuje naąe zadání.

Kapitola 2

Vymezení požadavků

Na základě oficálního zadání této práce, kterým začíná tento text, vymezíme v této kapitole požadavky na implementovaný software. Učiníme tak nejen proto, abychom mohli navrhnout všechny potřebné součásti systému, ale i proto, abychom byli schopni po provedení implementace ověřit, zdali jsme tyto požadavky splnili.

2.1 Funkční požadavky

Funkční požadavky přímo popisují cíle, kterých má projekt (v našem případě program) dosáhnout. Na základě funkčních požadavků lze navrhnout metody testování a ověřit, zdali bylo zadání úspěšně splněno.

2.1.1 Manuál

Součástí zadání práce je uživatelský manuál. Nefunkčním požadavkem na manuál je jeho forma ve standardní manuálové stránky, které tvoří základ dokumentace v unixových operačních systémech.

2.2 Nefunkční požadavky

Mezi nefunkční požadavky řadíme implementační náležitosti, které popisují způsob, jakým máme implementaci provést. Některé z nich jsou součástí zadání, některé si musíme stanovit sami. Nefunkční požadavky nejsou předmětem ověřování, protože tvoří předpoklady, ze kterých vycházíme.

2.3 Manuál

Součástí zadání práce je uživatelský manuál. Nefunkčním požadavkem na manuál je jeho forma ve standardní manuálové stránky.

2.4 Dokumentace

Vytvorte uživatelský manuál ve forme standardní man-page, jakož i programátorskou dokumentaci. Využijte existujících nástroju UNIXu.

2.5 Testování

Dle zadání práce je třeba otestovat, zdali program korektně zpracovává korektní vstup, a korektně odmítá vstup nekorektní. Metody testování zdokumentujeme později v příslušné kapitole. Testování provádíme na základě stanovených funkčních požadavků.

Kapitola 3

Analýza a návrh

Kapitola 4

Implementace

Kapitola 5

Testování

Kapitola 6

Rozšířitelnost

Důkaz z předpokladů, další pravidla (Gentzen,...) namísto MP / vzájemný převod

Závěr

Literatura

[1] Sochor, A.: *Klasická matematická logika*. Praha: Univerzita Karlova v Praze, 2001, ISBN 80.

[2] Švejdar, V.: *Logika: neúplnost, složitost a nutnost*. Praha: Academia, 2002, ISBN 80.

Příloha **A**

Obsah přiloženého CD

- `src` ..... adresář se zdrojovou formou práce
  - `impl` ..... adresář se zdrojovou formou implementace
    - `doc` ..... adresář se soubory programátorské dokumentace
    - `src` ..... adresář se soubory zdrojového kódu
    - `test` ..... adresář se soubory pro testování
    - `Doxyfile` ..... soubor konfigurace dokumentace Doxygen
    - `mainpage.dox` soubor hlavní stránky programátorské dokumentace
    - `Makefile` ..... soubor konfigurace sestavení programu
    - `pl.1` ..... soubor zdrojové formy manuálové stránky
    - `test.sh` ..... soubor testovacího skriptu pro shell
  - `thesis` ..... adresář se zdrojovou formou textu práce
    - `bibliography.bib` ..... soubor bibliografických zdrojů
- `text` ..... adresář s textem práce

Příloha B

Instalační příručka