Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Softwarová podpora výuky předmětu Matematická logika

Aplikace je určena pro praktické procvičování základů výrokové logiky

2018

Radek Lipenský

Vedoucí práce: Doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Radek Lipenský
Název práce:	Softwarová podpora výuky předmětu Matematická logika (Aplikace je určena pro praktické procvičování základů výrokové logiky)
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2018
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	Doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	43
Přílohy:	CD vytvořené
Jazyk práce:	český

Bibliograhic info

Author:	Radek Lipenský
Title:	Software support teaching of the subject Mathematical logic (The application is intended for practicing basic principles of propositional calculus)
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2018
Study field:	Applied Computer Science, combined form
Supervisor:	Doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	43
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Výsledkem mé bakalářské práce, jejíž stěžejní část tvoří přiložená aplikace, je určen především pro podporu výuky předmětu Matematická logika, vyučovanému na Přírodovědecké fakultě Univerzity Palackého v Olomouci. Daná aplikace se nesoustředí na teoretické základy, které je třeba nastudovat z jiných prací, ale poskytuje uživateli možnost si dané znalosti procvičit na příkladech. Nenahrazuje tedy klasické učebnice, ale je jejich doplňkem. Umožňuje uživateli vkládat vlastní formule výrokové logiky nebo tabulky pravdivostních hodnot a tyto dále analyzovat. Dovoluje posoudit sémantické vyplývání a sémantickou ekvivalenci, převést formuli do normálních forem nebo na formule tvořené pouze bázovými spojkami, posoudit splnitelnost formulí a další. Její součástí je i malý test znalostí. Důraz je kladen především na funkčnost a srozumitelnost aplikace, a zároveň se také snaží poskytnout uživateli maximální množství informací. Tato aplikace byla vytvořena ve vývojovém prostředí MS Visual Studio 2010 a pro svůj běh vyžaduje .NET Framework 4.7. Odladěna byla na počítači s operačním systémem Windows 7 Professional SP1. Aplikace je ve formě spustitelného souboru a nevyžaduje instalaci.

Synopsis

The result of my Bachelor's Thesis, the core of which is the attached application, is intended primarily to support the teaching of Mathematical Logic, taught at the Faculty of Science of the Palacký University in Olomouc. The given application does not focus on the theoretical basics that need to be studied from other works, but provides the user with the ability to practise the given knowledge on examples. It does not replace classical textbooks, but it is their complement. It allows the user to insert custom formulas of propositional calculus or truth table to analyse them. It allows assessing semantic deduction and semantic equivalence, converting formulas to normal forms, or formulas made up of base connectives only, evaluating formula fulfilment, and more. It also includes a small test of knowledge. Emphasis is put on the functionality and clarity of the application, as well as trying to provide the user with the maximum amount of information. This application was created in MS Visual Studio 2010 development environment and requires .NET Framework 4.7 for its execution. It was debugged on a computer running Windows 7 Professional. The application is in the form of an executable program and does not require installation.

Klíčová slova: výroková logika; bázové spojky; normální formy formulí; sémantické vyplývání; sémantická ekvivalence; logické funkce

Keywords: propositional calculus; base connectives; normal forms of formulas; semantic results; semantic equivalence; truth functions

Děkuji všem, kteří mi umožnili napsat tuto práci. Protože největší problémy byly s časem, děkuji zejména Charitě Letohrad a jejím asistentkám. Dále bych chtěl poděkovat také všem lidem na katedře informatiky, kteří se snažili vyjít mi vstříc vzhledem k mé situaci. V neposlední řadě děkuji též Doc. RNDr. Miroslavu Kolaříkovi, Ph.D. za vysvětlení některých pojmů a další rady a připomínky a také všem dalším lidem, kteří přispěli radou či pomocí k dokončení páce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

1.1 Úvodní slovo

Jak již bylo konstatováno v abstraktu, aplikace byla navržena na podporu výuky předmětu Matematická logika. Předmět s podobným obsahem je jedním z těch, které se učí na všech školách, vyučujících informatiku nebo matematiku. Jeho pochopení je základním kamenem pro pochopení další látky, a proto jsem se rozhodl vybrat si z navržených témat právě toto téma. Protože však téma matematické logiky je velmi obsáhlé, pro zpracování jsem si vybral pouze jednu malou část matematické logiky, a to výrokovou logiku.

1.2 Základní funkce aplikace:

Aplikace je navržena tak, aby byl schopna následujících činností:

- umožní zadat výrokovou formuli s maximálně čtyřmi výrokovými symboly
- rozpozná a nepřijme chybně zadanou formuli
- alternativně umožní zadat tabulku pravdivostních hodnot se čtyřmi výrokovými symboly
- pomocí Karnaughových map převede zadanou tabulku na zápis pomocí symbolů používaného jazyka VL
- převede zadanou formuli na úplnou konjunktivní nebo disjunktivní normální formu
- převede zadanou formuli na bázové spojky (5 bází)
- určí u zadané formule splnitelnost a posoudí, zda se jedná o tautologii nebo kontradikci
- určí, zda je daná formule sémanticky ekvivalentní s jinou formulí
- určí, zda daná formule sémanticky vyplývá z dané množiny formulí (až čtyři formule)
- je schopna zpracovat a vytvořit tabulku pravdivostních hodnot i pro formule v jazyce, obsahujícím symboly reprezentující běžně neužívané logické spojky
- krátký test znalostí, v níž použité formule a tabulky mají hodnoty generovány počítačem

1.3 Předchozí řešení

Práce, řešící dané téma, byla již na katedře informatiky zpracována v roce 2014 Mgr. Ivou Navrátilovou. Jí předložená aplikace se více zaměřuje na teoretickou stránku, zatíco procvičovat jdou pouze základní operace. Používá také efektnější grafický styl. Já jsem pojal tuto práci opačně a více než teorii se věnuji možnosti procvičování základních operací. Moje práce je tedy více zaměřena na fukcionalitu (Karnaughovy mapy, vyplývání z více formulí,...), což předchozí práce neřešila. Grafické řešení mé práce také více vychází ze zásad pro tvorbu GUI na počítačích se systémem Windows. [1]

2 Syntaxe výrokové logiky

2.1 Co je logika

Logika je základ pro formální metody v informatice. Vztah mezi logikou a informatikou je velmi úzký. Logika se používá pro analýzy dat nebo při sestavování logických obvodů. Ale co je vlastně logika? Je to v první řadě věda zabývající se správným usuzováním. Ale v logice jde o formu tohoto usuzování, ne o jeho obsah. Proto se moderní logika obvykle nazývá logika formální(symbolická). Za zakladatele logiky je obvykle pokládán Aristoteles (384 - 322 př. n. l.) Velký rozvoj matematické logiky nastal v 19. a 20. století, logika se začíná formalizovat V dnešní době je znalost logiky důležitou součástí vzdělání každého informatika. Pro něj je důležité umět svoje požadavky a návrhy sdělit počítači, a toto sdělení musí být přesné. Další části logiky jsou pak nepostradatelné pro analýzu dat nebo různé expertní systémy.

2.2 Různé logiky

Jak již bylo uvedeno, v logice jde o formu, ne o obsah. Tato formalizace umožňuje, abychom se dokázali vyhnout různým logickým paradoxům, jako je například známý paradox lháře („V tomto okamžiku lžu“). Pokud ke studiu logiky používáme matematických metod, obvykle ji označujeme matematickou logikou. Dále logiku běžně členíme na

- **klasickou logiku** - logika, kde tvrzení mohou nabývat pouze dvou pravdivostních hodnot a pravdivostní hodnota složených tvrzení závisí na pravdivostních hodnotách skládaných tvrzení
- **neklasickou logiku** - tvrzení mohou nabývat více pravdivostních hodnot, neklasické spojky, ...
  - **modální logika** - spojky jako „je možné, že ...“
  - **epistemická logika** - spojky jako „ví se, že ...“
  - **temporální logika** - tvrzení, kde svou roli hraje i čas

– **fuzzy logika** – více pravdivostních hodnot

Předkládaná aplikace se zaměřuje na jednu část klasické logiky, a to logiku výrokovou. Ta z pohledu formalizace zkoumá usuzování na úrovni vět v souvětích na rozdíl od logiky predikátové, která zkoumá usuzování až na úroveň větných členů.

2.3 Výroková logika

Výrok je tvrzení, o němž má smysl uvažovat, zda je pravdivé či nikoliv (např. „Dnes jsem byl celý den doma."). Z takto jednoduchých výroků tvoříme složitější pomocí logických spojek. To jsou speciální jazykové výrazy jako „... a ...", „jestliže ..., pak ..." a další.

2.4 Jazyk výrokové logiky

Definice 1 (Definice jazyka VL)
Jazyk výrokové logiky se skládá z

- **výrokových symbolů** obvykle označovaných p, q, r, ..., případně s indexy p_1, p_2, p_3 ...
- **symbolů výrokových spojek:**
  - $\neg$ negace
  - $\Rightarrow$ implikace
  - $\wedge$ konjunkce
  - $\vee$ disjunkce
  - $\Leftrightarrow$ ekvivalence
  - $\oplus$ nonekvivalence
  - $\Leftarrow$ zpětná implikace
  - $\not\Rightarrow$ inhibice
  - $\not\Leftarrow$ zpětná inhibice
  - $\Uparrow$ Sheffer
  - $\Downarrow$ Pierce (Nicod)
- **pomocných symbolů** – v této aplikaci jsou využívány pouze kulaté závorky (,).

Tato aplikace standardně využívá jazyk výrokové logiky složený ze spojek odpovídajících logickým funkcím negace (NOT), implikace, ekvivalence, konjunkce(AND – logický součin) a disjunkce(OR – logický součet). Veškeré převody jsou také realizovány z tohoto jazyka VL. Pokud se jedná o zpracování formule zadané pomocí symbolů jazyka, je možné použít i jazyk VL obsahující další výrokové spojky. Jde o spojky odpovídající logickým funkcím nonekvivalence(pro dvouargumentové funkce

odpovídá výlučné disjunkci XOR), inhibice, zpětná inhibice, zpětná implikace, negovaný logický součin (NAND - Shefferova spojka) a negovaný logický součet(NOR - Piercova(Nicodova) spojka).

2.5 Formule výrokové logiky

Definice 2 (Definice formule VL)

Nechť je dán jazyk VL. Formule(daného jazyka) výrokové logiky je definována následovně:

- každý výrokový symbol je formule (atomická formule)
- jsou-li φ a ψ formule, pak i
  - $\neg\varphi$
  - $(\varphi \Rightarrow \psi)$
  - $(\varphi \wedge \psi)$
  - $(\varphi \vee \psi)$
  - $(\varphi \Leftrightarrow \psi)$
  - $(\varphi \oplus \psi)$
  - $(\varphi \Leftarrow \psi)$
  - $(\varphi \not\Rightarrow \psi)$
  - $(\varphi \not\Leftarrow \psi)$
  - $(\varphi \Uparrow \psi)$
  - $(\varphi \Downarrow \psi)$

jsou formule daného jazyka VL.

Jak již bylo uvedeno výše, pro potřeby této aplikace i dalšího výkladu budeme považovat za formule používaného jazyka VL pouze prvních pět zápisů formulí.

Pro zpřehlednění zápisu formulí budeme používat konvenci o vynechání vnějších závorek. Pro možnost vynechání vnitřních závorek budeme uvažovat prioritu symbolů výrokových spojek následovně: [2]

1. $\neg$
2. $\wedge$, $\Uparrow$
3. $\vee$, $\Downarrow$
4. $\Rightarrow$, $\Leftarrow$, $\not\Rightarrow$, $\not\Leftarrow$
5. $\Leftrightarrow$, $\oplus$

Tabulka 1: Logické operace

	$\neg$
0	1
1	0

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\Rightarrow$	0	1
0	1	1
1	0	1

$\Leftrightarrow$	0	1
0	1	0
1	0	1

Aplikace tuto konvenci plně podporuje. Budeme tedy za formule považovat i $\varphi \wedge \psi$ nebo $\varphi \Rightarrow \psi \vee \varphi$, kde u druhé formule bude nejdříve provedena disjunkce a poté implikace. Při striktním dodržení definice by toto nebyly formule. Někdy je třeba odkazovat na části formulí, které jsou sami také formulemi. Máme-li formuli φ, pak každá formule, která se nachází uvnitř φ jako podřetězec se nazývá podformule φ.

3 Sémantika výrokové logiky

3.1 Pravdivost formulí

Zatím jsme se věnovali formulím pouze z pohledu *syntaxe*. Ta nedefinuje pravdivost formulí. Formule samy nemají žádný význam. Přiřazení významu syntaktickým objektům se věnuje *sémantika*. Nejdříve si definujme pravdivostní ohodnocení.

Definice 3 (Definice pravdivostního ohodnocení)
Pravdivostní ohodnocení je libovolné zobrazení e výrokových symbolů daného jazyka výrokové logiky do množiny {0,1}. Ohodnocení e přiřazuje každému výrokovému symbolu p hodnotu 0 nebo 1, kde **0** reprezentuje pravdivostní hodnotu **nepravda** a **1** hodnotu **pravda**.

Definice 4 (Pravdivostní hodnota formule daného jazyka VL)
Nechť je dáno pravdivostní ohodnocení e – toto pravdivostní ohodnocení musí být dáno „zvenčí". Pak pravdivostní hodnota formule φ při ohodnocení e, kterou značíme $||\varphi||_e$ je definována následovně:

- jestliže je φ výrokovým symbolem p, pak je $||p||_e = \mathrm{e(p)}$
- jestliže je φ složená formule, t.j. obsahuje logické spojky, pak pravdivostní hodnotu výroku $||\varphi$ „symbol logická spojka" $\psi||_e$ získáme použitím pravdivostní funkce reprezentované daným symbolem logické spojky na pravdivostní hodnoty $||\varphi||_e$ a $||\psi||_e$.

Je-li $||\varphi||_e = 1$ ($||\varphi||_e = 0$) pak říkáme, že formule φ je při ohodnocení e pravdivá (nepravdivá). Je tedy zřejmé, že pravdivost formule chápeme vždy vzhledem k nějakému ohodnocení. Pravdivostní funkce(operace) jednotlivých logických spojek nejpřehledněji zobrazíme pomocí tabulek pravdivostních hodnot při všech ohodnoceních, jak ukazuje tabulka1.

Problémem je, že v jazyku VL předpokládáme *nekonečně* mnoho výrokových symbolů. Oproti tomu se v každé formuli vyskytuje pouze *konečně* mnoho výrokových symbolů. Zavedeme proto výraz $\varphi(p_1, \ldots, p_n)$, kde φ je formule jazyka VL a $p_1, \ldots, p_n$ jsou právě všechny výrokové symboly vyskytující se ve formuli φ a platí, že pravdivost formule φ při daném pravdivostním ohodnocení závisí *pouze* na ohodnocení výrokových symbolů, které se vyskytují ve formuli φ.

3.2 Tabulková metoda

Základem tabulkové metody je úvaha, že pro n výrokových symbolů $p_1, \ldots, p_n$ existuje právě 2^n různých ohodnocení výrokových symbolů $p_1, \ldots, p_n$, kde je každému výrokovému symbolu přiřazena hodnota 0 nebo 1. Tabulkovou metodu můžeme použít k tabelaci více formulí například při dokazování sémantického vyplývání nebo i pro zjišťování pravdivostní hodnoty formule složené z více podformulí. Je-li n počet výrokových symbolů obsažených ve formuli φ (nebo ve formulích $(\psi_1, \ldots, \psi_m)$, pak má tabulka 2^n řádků a $n + m$ sloupců, kde m znamená počet všech vlastních podformulí formule φ (nebo počet formulí $\psi_1, \ldots, \psi_m$). Řádky pak reprezentují pravdivostní ohodnocení symbolů a sloupce pravdivostní hodnoty podformulí formule φ (nebo pravdivostní hodnoty formulí $\psi_1, \ldots, \psi_m$).

3.3 Sémantické vyplývání a ekvivalence

Intuitivní pojem *vyplývání* definujeme následovně:

Definice 5 (Sémantické vyplývání)
Formule ψ **sémanticky plyne** z formule φ (značíme $\varphi \vDash \psi$), jestliže formule ψ je pravdivá při každém ohodnocení, při kterém je pravdivá formule φ. Obecněji: formule ψ **sémanticky plyne z množiny formulí** T (značíme $T \vDash \psi$), je-li formule ψ pravdivá při **každém** ohodnocení, při kterém je pravdivá **každá** formule z T. Formule z množiny T nazýváme sémantické **předpoklady**, formuli ψ nazýváme sémantický **důsledek**.

Související pojem sémantická ekvivalence ja definován následovně:

Definice 6 (Sémantická ekvivalence)
Pokud formule ψ sémanticky plyne z formule φ **a naopak**, říkáme, že formule ψ a φ jsou **sémanticky ekvivalentní**.

Zřejmě tedy formule φ, ψ jsou sémanticky ekvivalentní, pokud $||\varphi||_e = ||\psi||_e$ **pro každé** ohodnocení e. Tedy sémanticky ekvivalentní formule od sebe nelze rozlišit pravdivostí.

3.4 Zákony VL

Formule VL se nazývá:

- **tautologie**, je-li při každém ohodnocení pravdivá (píšeme $\vDash \varphi$ nebo $||\varphi|| = 1$)
- **kontradikce**, je-li při každém ohodnocení nepravdivá
- **splnitelná**, je-li pravdivá alespoň při jednom ohodnocení

Některé tautologie povyšujeme na tzv. **zákony VL:**
Výběr ze zákonů VL:

- zákon vyloučeného třetího: $\varphi \vee \neg\varphi$
- zákon sporu: $\neg(\varphi \wedge \neg\varphi)$
- zákon dvojí negace: $\neg\neg\varphi \Leftrightarrow \varphi$
- de Morganův zákon 1: $\neg(\varphi \wedge \psi) \Leftrightarrow (\neg\varphi \vee \neg\psi)$
- de Morganův zákon 2: $\neg(\varphi \vee \psi) \Leftrightarrow (\neg\varphi \wedge \neg\psi)$
- náhrada implikace: $(\varphi \Rightarrow \psi) \Leftrightarrow (\neg\varphi \vee \psi)$
- náhrada ekvivalence: $(\varphi \Leftrightarrow \psi) \Leftrightarrow ((\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi))$

Výběr ze zákonů VL je volen tak, aby co nejvíce přispěl k pochopení převodů na bázové spojky (viz. dále).

3.5 Booleovské funkce

Definice 7 (Booleovská funkce s n argumenty)
Booleovská funkce s n argumenty (n-ární booleovská funkce) je libovolné zobrazení, které každé **uspořádané n-tici hodnot** 0 nebo 1 přiřadí **hodnotu** 0 nebo 1.

Booleovskou funkci f můžeme také zapsat: $\{0,1\}^n \to \{0,1\}$. Každou booleovskou funkci můžeme zapsat v tabulce (podobně jako u tabulkové metody). Předpokládejme, že argumenty funkce f s n argumenty označíme $x_1, \ldots, x_n$, pak píšeme také $f(x_1, \ldots, x_n)$. Všechny booleovské funkce jedné a dvou proměnných vidíme v tabulkách 2 a 3. Z tabulek je vidět, že například f_2 jedné proměnné je pravdivostní funkcí výrokové spojky negace. Výrokové spojky, jejichž pravdivostní funkce odpovídá dané booleovské funkci, jsou v tabulce 3 v posledním řádku.

Věta 8
Existuje $2^{(2^n)}$ booleovských funkcí s n argumenty.

Protože funkce mají n argumentů, má příslušná tabulka 2^n řádků. Protože v každém řádku je jedno volné místo pro hodnotu funkce, volných míst je tedy také 2^n. Protože každé lze vyplnit buď 0 nebo 1, lze tabulku vyplnit právě $2^{(2^n)}$ způsoby. Platí

Tabulka 2: Booleovské funkce jedné proměnné

x_1	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Tabulka 3: Booleovské funkce dvou proměnných

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
		$\mathcal{F}$	$\wedge$	$\not\Rightarrow$	x_1	$\not\Leftarrow$	x_2	$\oplus$	$\vee$

x_1	x_2	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
		$\Downarrow$	$\Leftrightarrow$	$\neg x_2$	$\Leftarrow$	$\neg x_1$	$\Rightarrow$	$\Uparrow$	$\mathcal{V}$

ale i opačné tvrzení: Ke každé booleovské funkci f s n argumenty existuje formule φ_f taková, že tato formule indukuje právě danou funkci f. Platí dokonce, že formule φ_f může obsahovat pouze spojky $\neg, \wedge, \vee$.

3.6 Úplné systémy spojek

3.6.1 Normální formy formulí VL

Ke každé formuli výrokové logiky je jednoznačně přiřazena pravdivostní funkce, ale k dané funkci existuje mnoho formulí výrokové logiky, které ji mají za svou. Definujeme proto standardní (kanonické) tvary formulí výrokové logiky, kdy každá třída navzájem ekvivalentních formulí bude reprezentována jedinou formulí ve standardním tvaru.

Definice 9 (Standardní tvary formulí)
Nechť V je množina výrokových symbolů. Pak

- **literál** nad V je libovolný výrokový symbol z V nebo jeho negace
- **elementární konjunkce (EK)** nad V je libovolná konjunkce literálů
- **elementární disjunkce (ED)** nad V je libovolná disjunkce literálů
- **úplná elementární konjunkce (ÚEK)** nad V je libovolná konjunkce literálů, ve které se každý výrokový symbol z V vyskytuje právě v jednom literálu

- **úplná elementární konjunkce (ÚED)** nad V je libovolná disjunkce literálů, ve které se každý výrokový symbol z V vyskytuje právě v jednom literálu
- **disjunktivní normální forma (DNF)** dané formule nad V je formule ekvivalentní s danou formulí a mající tvar disjunkce elementárních konjunkcí
- **konjunktivní normální forma (KNF)** dané formule nad V je formule ekvivalentní s danou formulí a mající tvar konjunkce elementárních disjunkcí
- **úplná disjunktivní normální forma (ÚDNF)** dané formule nad V je formule ekvivalentní s danou formulí a mající tvar disjunkce úplných elementárních konjunkcí
- **úplná konjunktivní normální forma (ÚKNF)** dané formule nad V je formule ekvivalentní s danou formulí a mající tvar konjunkce úplných elementárních disjunkcí

Tyto definice jsou převzty z [3], ale lze je nalézt takřka ve všech zde citovaných zdrojích.

Věta 10

Ke každé formuli VL, která není tautologií (kontradikcí) existuje s ní sémanticky ekvivalentní formule, která je ve tvaru úplné konjunktivní normální formy (úplné disjunktivní normální formy).

Konstrukce ÚDNF pro formuli φ s výrokovými symboly $p_1, \ldots, p_n$:

1. pro formuli $\varphi(p_1, \ldots, p_n)$ sestrojíme tabulku pravdivostních hodnot
2. pro řádky s hodnotou 1 ve sloupci odpovídajícímu pravdivostní hodnotě formule φ sestrojíme ÚEK z p_i pro hodnotu 1 a $\neg p_i$ pro hodnotu 0
3. výsledná ÚDNF je konjunkcí takových ÚEK

Konstrukce ÚKNF pro formuli φ s výrokovými symboly $p_1, \ldots, p_n$:

1. pro formuli $\varphi(p_1, \ldots, p_n)$ sestrojíme tabulku pravdivostních hodnot
2. pro řádky s hodnotou 0 ve sloupci odpovídajícímu pravdivostní hodnotě formule φ sestrojíme ÚED z p_i pro hodnotu 0 a $\neg p_i$ pro hodnotu 1
3. výsledná ÚKNF je konjunkcí takových ÚED

Výsledek si můžeme zkontrolovat třeba v předkládané aplikaci.

Tabulka 4: Karnaughova mapa

p	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
q	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
r	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
s	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$\|\|\varphi\|\|$	0	0	0	1	0	0	1	1	1	0	0	1	0	0	1	1

			r	r	
				s	s
		0	0	1	0
p		0	1	1	0
p	q	0	1	1	0
	q	1	0	1	0

3.6.2 Minimalizace logické funkce pomocí Karnaughovy mapy

Jak již bylo výše uvedeno, k dané logické funkci existuje mnoho formulí, které ji mají za svou. Pro mnoho aplikací (např.: navrhování logických obvodů pomocí hradel) je výhodné využít formuli odpovídající logické funkci v minimalizované podobě. Pro tuto minimalizaci je často používáno zobrazení pomocí Karnaughových map, zejména pokud daná logická funkce přijímá maximálně čtyři vstupy (má čtyři vstupní proměnné), t.j. odpovídá formuli s maximálně čtyřmi výrokovými symboly, které zde pro jednoduchost budeme značit p, q, r, s. Stejně jako při konstrukci normálních forem i zde lze používat formu disjunktivní (sjednocují se logické 1) nebo konjunktivní (sjednocují se logické 0). V praxi se běžně používá forma disjunktivní. Má-li logická funkce n vstupních proměnných, rozdělíme je na přibližně stejné skupiny obsahující n_1 a n_2 proměnných (n = $n_1 + n_2$). Pak sestavíme mapu, která bude obsahovat 2^{n_1} sloupců a 2^{n_2} řádků, to znamená celkem $2^{n_1+n_2} = 2^n$ polí. Mapa samotná je vlastně transformací pravdivostní tabulky, kde každému políčku mapy přiřadíme jednu z kombinací vstupních proměnných tak, že v každém políčku mapy je zapsána hodnota logické funkce (0 nebo 1), která představuje pravdivostní hodnotu funkce pro logické proměnné příslušného řádku a sloupce. V hlavičce tabulky se obvykle označují řádky a sloupce, pro které je logická hodnota dané proměnné rovna 1. Pro představu je příklad v tabulce 4.

Další postup je následující:

- V Karnaughově mapě označíme podmapy(smyčky)
- vybrané smyčky musí pokrývat všechny jedničkové (pro disjunktivní formu) nebo nulové (pro konjunktivní formu) stavy
- do smyčky spojujeme stejné stavy sousedící hranou, a to i přes okraj mapy

(t.j. u tabulky se 16 políčky 4 x 4 sousedí čtvrtý řádek s prvním a také čtvrtý sloupec s prvním). Rohy mapy jsou též sousedními prvky

- smyčka musí mít tvar pouze čtverce nebo obdélníku, počet jejichž členů je roven mocninám 2 (t.j. 1, 2, 4, 8, ...)

- smyčky se snažíme vytvářet co největší - smyčky se mohou prolínat, ale nevytváříme zbytečné smyčky (t.j. takové, že všechny členy dané smyčky jsou už obsaženy ve větších smyčkách).

Pro každou smyčku sestavíme potom elementární konjunkci (nebo disjunkci), která obsahuje pouze symboly(proměnné), které v dané smyčce nemění hodnotu (t.j. v celé smyčce mají hodnotu 0 nebo 1). U konjunkcí bereme pro hodnotu 1 symboly p, q, r, s a pro hodnotu 0 jejich negace (¬p, ¬q, ¬r, ¬s), u disjunkcí potom pro hodnotu 0 symboly p, q, r, s a pro hodnotu 1 jejich negace (¬p, ¬q, ¬r, ¬s). Výsledná formule je pak disjunkcí uvedených konjunkcí, případně konjunkcí uvedených disjunkcí.

Další a také podrobnější vysvětlení lze nalézt např. v [4], ale také v [5], [6]

3.6.3 Úplné systémy spojek

Připomeňme si definici booleovské funkce: Booleovská funkce(n-ární) je libovolné zobrazení f: $\{0,1\}^n \rightarrow \{0,1\}$. Každému symbolu logické spojky odpovídá booleovská funkce, např. symbolu $\wedge$ odpovídá funkce logický součet(AND).

Definice 11 (Funkčně úplná množina booleovských funkcí)

Množina booleovských funkcí $\{f_1, \ldots, f_k\}$ je **funkčně úplná**, pokud každou booleovskou funkci f: $\{0,1\}^n \rightarrow \{0,1\}$ lze vyjádřit jako složení některých funkcí z $\{f_1, \ldots, f_k\}$.

Definice 12 (Úplný systém spojek)

Množina výrokových spojek je **úplná** (tvoří **úplný systém spojek**), jestliže je funkčně úplná množina jim odpovídajících booleovských funkcí.

Říkáme také, že množina logických spojek T tvoří **úplný systém spojek** právě tehdy, když ke každé formuli φ existuje formule ψ taková, že formule φ, ψ jsou sémanticky ekvivalentní a formule ψ obsahuje pouze logické spojky z dané množiny T. Každý úplný **minimální** systém spojek VL nazýváme **bází**. Čím více výrokových spojek obsahuje množina všech spojek daného jazyka, tím je vyjádření formulí jednodušší, ale zároveň vzrůstá délka formulí. Například množina $\{\neg, \wedge, \vee\}$ tvoří úplný systém spojek (platnost plyne z vět o ÚDNF, ÚKNF). Z de Morganových zákonů je zároveň zřejmé, že se nejedná o bázi. Dá se dokázat (pomocí zákonů VL), že existují dvouprvkové báze $\{\neg, \Rightarrow\}$, $\{\neg, \wedge\}$, $\{\neg, \vee\}$. Speciální význam mezi výrokovými spojkami mají Shefferova spojka (značíme symbolem $\Uparrow$ a odpovídá pravdivostní funkci NAND) a Piercova(Nicodova) spojka (značíme symbolem $\Downarrow$ a odpovídá pravdivostní funkci NOR). Tyto funkce mají následující tabulky pravdivostní hodnot, viz.tabulka 5

Tabulka 5: Logické operace NAND a NOR

$\uparrow$	0	1
0	1	1
1	1	0

$\downarrow$	0	1
0	1	0
1	0	0

Tyto spojky samy o sobě tvoří úplný systém spojek (a také tvoří jediné dvě jednoprvkové báze). Tedy pomocí těchto spojek lze nahradit všechny ostatní běžně užívané spojky ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$). Vycházíme z těchto základních převodů:

- $(\varphi \Uparrow \psi) \Leftrightarrow \neg(\varphi \wedge \psi)$
- $(\varphi \Downarrow \psi) \Leftrightarrow \neg(\varphi \vee \psi)$

Závěrem bych ještě uvedl převody ostatních výrokových spojek na výše uvedené báze, které jsou používané v představované aplikaci:

1. Báze tvořená výrokovou spojkou Sheffer $\{\Uparrow\}$:
   - $\neg\varphi \Leftrightarrow \neg(\varphi \wedge \varphi) \Leftrightarrow (\varphi \Uparrow \varphi)$
   - $(\varphi \wedge \psi) \Leftrightarrow \neg\neg(\varphi \wedge \psi) \Leftrightarrow \neg(\varphi \Uparrow \psi) \Leftrightarrow ((\varphi \Uparrow \psi) \Uparrow (\varphi \Uparrow \psi))$
   - $(\varphi \vee \psi) \Leftrightarrow \neg\neg(\varphi \vee \psi) \Leftrightarrow \neg(\neg\varphi \wedge \neg\psi) \Leftrightarrow (\neg\varphi \Uparrow \neg\psi) \Leftrightarrow ((\varphi \Uparrow \varphi) \Uparrow (\psi \Uparrow \psi))$
   - $(\varphi \Rightarrow \psi) \Leftrightarrow (\varphi \Uparrow (\varphi \Uparrow \psi))$
   - $(\varphi \Leftrightarrow \psi) \Leftrightarrow ((\varphi \Uparrow \psi) \Uparrow ((\varphi \Uparrow \varphi) \Uparrow (\psi \Uparrow \psi))$
2. Báze tvořená výrokovou spojkou Pierce (Nicod) $\{\Downarrow\}$:
   - $\neg\varphi \Leftrightarrow \neg(\varphi \vee \varphi) \Leftrightarrow (\varphi \Downarrow \varphi)$
   - $(\varphi \wedge \psi) \Leftrightarrow \neg\neg(\varphi \wedge \psi) \Leftrightarrow \neg(\neg\varphi \vee \neg\psi) \Leftrightarrow (\neg\varphi \Downarrow \neg\psi) \Leftrightarrow ((\varphi \Downarrow \varphi) \Downarrow (\psi \Downarrow \psi))$
   - $(\varphi \vee \psi) \Leftrightarrow \neg\neg(\varphi \vee \psi) \Leftrightarrow \neg(\varphi \Downarrow \psi) \Leftrightarrow ((\varphi \Downarrow \psi) \Downarrow (\varphi \Downarrow \psi))$
   - $(\varphi \Rightarrow \psi) \Leftrightarrow ((\varphi \Downarrow \psi) \Downarrow \psi) \Downarrow ((\varphi \Downarrow \psi) \Downarrow \psi)$
   - $(\varphi \Leftrightarrow \psi) \Leftrightarrow ((\varphi \Downarrow \psi) \Downarrow \psi) \Downarrow ((\varphi \Downarrow \psi) \Downarrow \varphi)$
3. Báze tvořená výrokovými spojkami negace a implikace $\{\neg, \Rightarrow\}$:
   - $(\varphi \wedge \psi) \Leftrightarrow \neg(\varphi \Rightarrow \neg\psi)$
   - $(\varphi \vee \psi) \Leftrightarrow \neg\varphi \Rightarrow \psi$
   - $(\varphi \Leftrightarrow \psi) \Leftrightarrow \neg((\varphi \Rightarrow \psi) \Rightarrow \neg(\psi \Rightarrow \varphi))$
4. Báze tvořená výrokovými spojkami negace a konjunkce $\{\neg, \wedge\}$:
   - $(\varphi \vee \psi) \Leftrightarrow \neg(\neg\varphi \wedge \neg\psi)$

- $(\varphi \Rightarrow \psi) \Leftrightarrow \neg(\varphi \wedge \neg\psi)$
- $(\varphi \Leftrightarrow \psi) \Leftrightarrow \neg(\varphi \wedge \neg\psi) \wedge \neg(\psi \wedge \neg\varphi)$

5. Báze tvořená výrokovými spojkami negace a disjunkce $\{\neg, \vee\}$:

   - $(\varphi \wedge \psi) \Leftrightarrow \neg(\neg\varphi \vee \neg\psi)$
   - $(\varphi \Rightarrow \psi) \Leftrightarrow \neg\varphi \vee \psi$
   - $(\varphi \Leftrightarrow \psi) \Leftrightarrow \neg(\neg(\neg\varphi \vee \psi) \vee \neg(\neg\psi \vee \varphi))$

Zde je použito některých příkladů z [7]. Jinak všechny definice a další teorie vycházejí ze skript [8], [9] a zejména ze slidů [10], odkud jsou i některé z příkladů poslední části.

4 Uživatelská dokumentace

4.1 Systémové požadavky

Aplikace je vytvořena tak, aby ji bylo možné spouštět na většině běžných stolních počítačů a notebooků, které využívají operační systém Windows a na nichž je nainstalováno rozhraní .NET Framework. Pro její bezproblémovou funkci je však doporučeno použít počítač s konfigurací, na kterém byla odladěna, t.j. počítač, na kterém bylo nainstalováno:

- Windows 7 Professional s SP1
- běhové prostředí .NET Framework 4.7

Při této konfiguraci je aplikace bez problémů funkční. Protože však ke své činnosti nevyžaduje žádné dodatečné knihovny a rozšíření, neměl by být problém ani s během pod vyššími(nižšími) verzemi operačního systému i běhového prostředí. To jsem neměl možnost vyzkoušet, ale je to velmi pravděpodobné. Pro plné zobrazení je nutné rozlišení monitoru minimálně 1050 x 900 pixelů. Při menším rozlišní už je nutné zmenšit hlavní okno, což lze, ale je to na úkor přehlednosti.

4.2 Základní informace o spuštění a běhu aplikace

Aplikace je ve formě spustitelného programu (.exe), a proto nevyžaduje žádnou instalaci. Spuštění je možné přímo otevřením souboru „Logika.exe“. Je pak možné si udělat zástupce na ploše nebo v nabídce „Start“. Protože jsem se při tvorbě uživatelského prostředí snažil dodržovat pravidla pro GUI programů na operačním systému Windows, mělo by být ovládání srozumitelné i naprostému laikovi. Daní za možnost rychlého výběru je pak větší množství tlačítek v hlavním okně aplikace, což ale myslím uživatel ne hned, ale po krátkém vyzkoušení, spíše ocení.

4.3 Základní funkce aplikace

Při zadání bakalářské práce bylo požadováno následující, což zároveň představuje hlavní funkce dané aplikace:

- umožní zadat výrokovou formuli s maximálně čtyřmi výrokovými symboly
- rozpozná a nepřijme chybně zadanou formuli
- alternativně umožní zadat tabulku pravdivostních hodnot se čtyřmi výrokovými symboly
- pomocí Karnaughových map převede zadanou tabulku na zápis pomocí symbolů používaného jazyka VL
- převede zadanou formuli na úplnou konjunktivní nebo disjunktivní normální formu
- převede zadanou formuli na bázové spojky (5 bází)
- určí u zadané formule splnitelnost a posoudí, zda se jedná o tautologii nebo kontradikci
- určí, zda je daná formule sémanticky ekvivalentní s jinou formulí
- určí, zda daná formule sémanticky vyplývá z dané množiny formulí (až čtyři formule)
- je schopna zpracovat a vytvořit tabulku pravdivostních hodnot i pro formule v jazyce, obsahujícím symboly reprezentující běžně neužívané logické spojky
- krátký test znalostí, v níž použité formule a tabulky mají hodnoty generovány počítačem

4.4 Popis jednotlivých částí aplikace

V následujícím textu jsou podrobněji popsány funkce aplikace. Pro přehlednost jsou řazeny stejně jako položky hlavního menu.

Po spuštění aplikace (popsáno výše) se objeví hlavní okno aplikace, viz. orb.1. Na horní liště záložek jsou tyto základní položky:

- Zadání
- Zobrazení
- Převody
- Nastavení
- Test

- Nápověda

Následující kapitoly teď popíší prostřednictvím popisu položek menu všechny funkce aplikace.

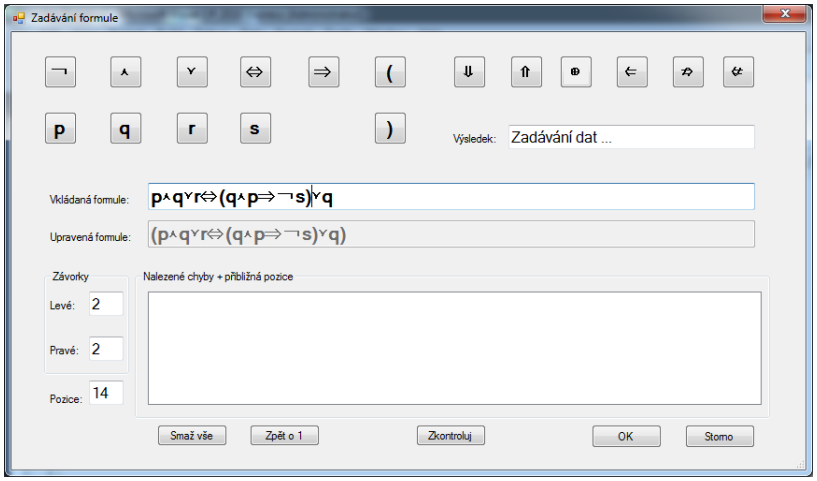
4.4.1 Zadání

Zde je možno zadávat zkoumané formule. Nabídka obsahuje tyto podnabídky:

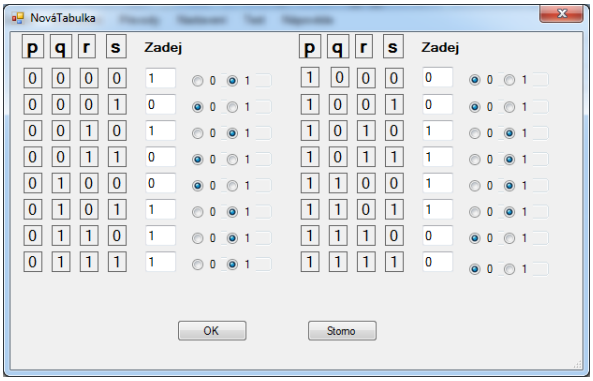
- **Zadat hlavní formuli** – (podnabídka „Zadat text formule“ nebo „Zadat tabulku“, u té je ještě podnabídka Symboly 1 až 4). Zde se zadává hlavní formule, ke které se vztahují všechny další funkce. Zadání formule bude popsáno níže.
- **Zadat další formule** – (podnabídka Formule 1 až 4). Zde je možno zadávat další formule, které jsou pak použity při posuzování sémantického vyplývání nebo sémantické ekvivalence. **Pokud není zadána ani jedna další formule, není možné použít tlačítka/funkce pro zobrazení sémantického vyplývání nebo ekvivalence.** Lze zadat až čtyři další formule.
- **Zavřít další formule** – pomocná položka pouze pro zavření panelu s dalšími formulemi. Panel se sám otevře, pokud zvolíte zadání dalších formulí.
- **Konec** – ukončení bežící aplikace.

Zadání formule. Pokud zvolíme zadání hlavní, případně dalších formulí, otevře se okno sloužící pro zadávání formule. Střední část okna tvoří textové pole „Vkládaná formule“. Zde se vkládají znaky buď z klávesnice, nebo výrokové spojky a další symboly pomocí zobrazených tlačítek. Pod ním je textové pole „Upravená formule“. Zde probíhá automatická konverze, spočívající v odstranění mezer a doplnění vnějších závorek. Ty jsou povinné, ale pokud je uživatel nezadá, doplní se automaticky . Vlevo dole je počítadlo závorek a ukazatel pozice kurzoru. Ve spodní části se potom zobrazují nalezené chyby a jejich přibližná pozice. Vzhledem ke způsobu vyhodnocování je těžké přesně určit pozici chyby a navíc chyba v hluboko vnořené pozici vyvolá i další chyby během návratu (pokud je chyba ve vnořené závorce, je chybná i nadřazená závorka). Formuli je možné otestovat tlačítkem „Zkontroluj“, zadání je ukončeno tlačítky „OK“ a „Storno“. druhé uvedené způsobí návrat bez zadání. Pro rychlejší práci a ovládání pomocí myši jsou zde i tlačítka „Smaž vše“ a „Zpět o 1“. V prvním případě dojde ke smazání celé formule, ve druhém případě je smazán znak vlevo od kurzoru. Příklad zadávání je na obrázku 1 Hlavní tabulku je možné zadat i pomocí tabulky pravdivostních hodnot. Nejdříve musíme z podmenu vybrat, pro kolik různých výrokových symbolů bude tabulka vytvořena. Potom se zobrazí okno pro zadání se všemi možnými kombinacemi výrokových symbolů. Do textových polí nebo pomocí přepínačů zvolíme naše pravdivostní hodnoty při všech různých ohodnoceních. Pomocí Karnaughovy mapy je pak aplikace převede na vyjádření pomocí odpovídající formule. Ukončení opět přes tlačítka „OK“ a „Storno“. Zadání je vidět na obrázku 2

Po zadání hlavní, případně dalších formulí, se nám zpřístupní další tlačítka. Tlačítka pro spuštění většiny funkcí jsou zdvojena. Většinu funkcí je tedy možné ovládat



Obrázek 1: Zadávání formule textově



Obrázek 2: Zadávání formule pomocí tabulky pravdivostních hodnot

jak z pásu nabídek v horní části, tak také pomocí tlačítek přímo na hlavním panelu. Z funkcí pro zadání jde o tlačítka pro zadání a smazání hlavní (nebo dalších) formulí.

U všech zadaných formulí je automaticky zjišťována splnitelnost a to, zda nejsou kontradikcemi nebo tautologiemi. Tyto údaje jsou zobrazeny vždy vedle dané formule, takže je dobrý přehled i o tom, jaké jsou formule, které slouží jako předpoklady k sémantickému vyplývání, případně k sémantické ekvivalenci.

4.4.2 Zobrazení

- **Zobrazit hlavní tabulku** – zobrazí tabulku pravdivostních hodnot hlavní formule.
- **Zobrazit další tabulky** – (podnabídka Tabulka 1 až 4) pro zobrazení tabulky pravdivostních hodnot dalších formulí.
- **Sémantické vyplývání** – zde je zjišťováno pomocí tabulkové metody, takže zobrazí příslušnou tabulku, kde jednotlivé sloupce odpovídají všem porovnávaným formulím. Výsledek je ještě zobrazen slovně na středním panelu. Zjišťování probíhá vždy ze všech formulí, t.j. zjišťujeme, zda hlavní formule sémanticky vyplývá ze **všech** zadaných dalších formulí. Řádky tabulky, které splňují předpoklad, tedy že při tomto pravdivostním ohodnocení je každá formule z předpokladů pravdivá, jsou barevně označeny. Pokud je pravdivý i důsledek (hlavní formule) je řádek označen žlutě, není-li pak zeleně. Vysvětlivky jsou dole pod tabulkou.
- **Sémantická ekvivalence** – (podnabídka Formule 1 až 4) Zde je možné si vybrat **již zadanou** formuli a porovnat, zda je sémanticky ekvivalentní s hlavní formulí. Jako u sémantického vyplývání je použita tabulková metoda a také zde jsou řádky splňující požadavek stejné pravdivostní hodnoty obarveny žlutě..

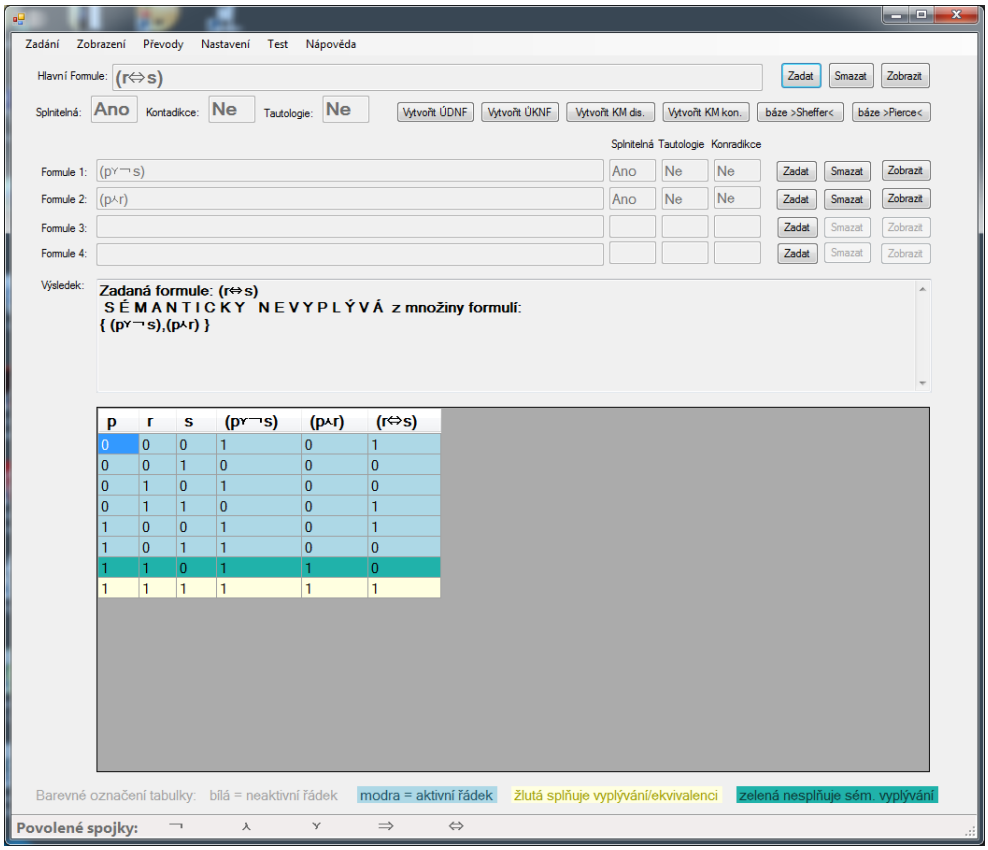
V této sekci je možné si zobrazit tabulky pravdivostních hodnot. Lze si zobrazit jak tabulku pravdivostních hodnot hlavní formule při všech možných pravdivostních ohodnoceních, tak také tyto tabulky pro další zadané formule. K tomu slouží buď položka „Zobrazení“ a její podpoložky „Zobrazit hlavní tabulku“ a „Zobrazit další tabulky“. Zde ještě musíme vybrat, tabulku které formule chceme zobrazit. Pro rychlejší práci potom slouží tlačítka na hlavním panelu, která mají opět stejnou funkci, jako položky z hlavní nabídky.

Zjišťování sémantického vyplývání a sémantické ekvivalence je možné spouštět opět i tlačítky na hlavním panelu, ale musí být zapnuto jejich zobrazení (viz. „Nastavení“, protože je zde standardně zobrazeno již dosti velké množství tlačítek a tyto funkce lze navíc používat pouze při zadání alespoň jedné další formule – mají tedy trochu specifické postavení. Příklad je na obrázku 3.

4.4.3 Převody

sekce je tvořena následujícími položkami:

- Vytvořit ÚKNF



Obrázek 3: Sémantické vyplývání

- Vytvořit ÚDNF
- KM disjunkce
- KM konjunkce
- Převod na bázi >Sheffer<
- Převod na bázi >Pierce<
- Převod na bázi Negace, implikace
- Převod na bázi Negace, konjunkce
- Převod na bázi Negace, disjunkce

Zaprvé je třeba zmínit, že převody na formule tvořené pouze symboly dané báze lze provádět **pouze** u formulí, zadaných pomocí čtveřice „klasických" výrokových spojek, t.j. implikací, ekvivalencí, konjunkcí a disjunkcí a také pomocí negace. Převody z formulí tvořených jinými spojkami nejsou dovoleny.

Teorie k tomuto tématu je dostatečně známá a pokud ne, je spousta literatury k nastudování dané problematiky, a to i pro zájemce s hlubším zájmem o dané téma. Pro základní pochopení pak stačí i text této bakalářské práce (úvodní sekce s teorií).

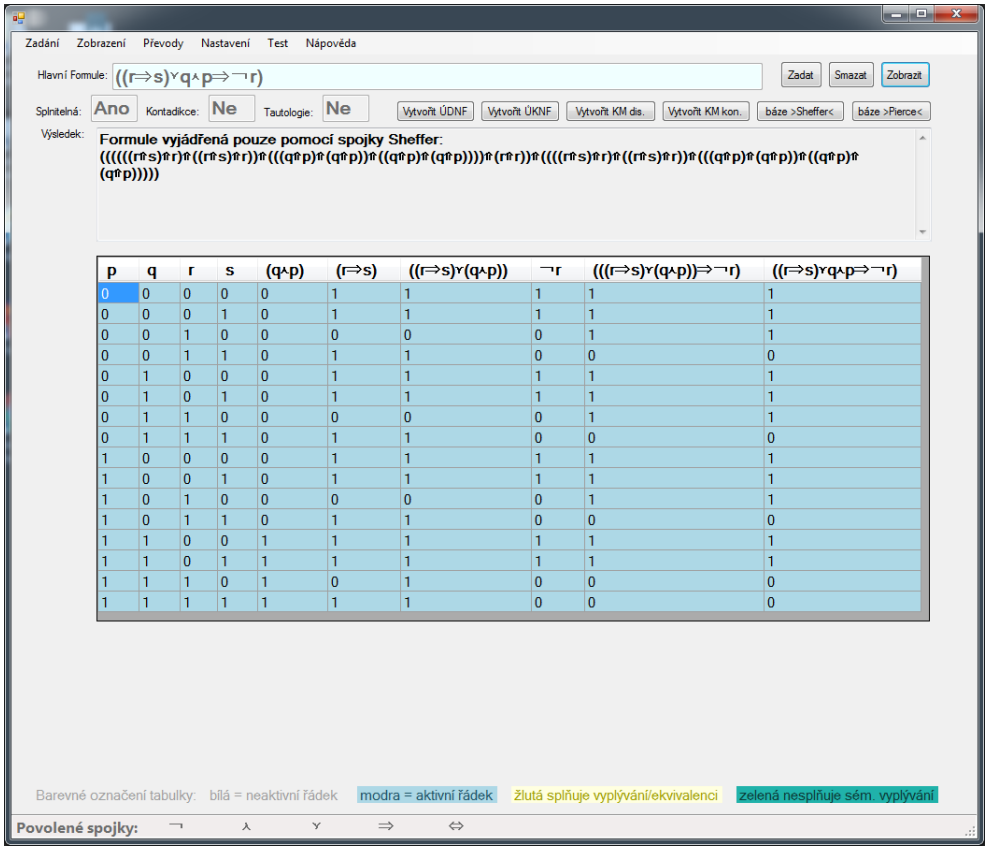
První dvě volby slouží k vytváření Úplné konjunktivní a Úplné disjunktivní normální formy hlavní formule. K vytvoření je použita tabulková metoda (teorie vytváření opět popsána výše v teoretické části). Tabulka pravdivostních hodnot použitá k tvorbě normálních forem je zobrazena a formule v podobě dané úplné normální formy je zobrazena v okně „Výsledek". Příklad ja na obrázku 4.

Podobně je řešena i minimalizace hlavní funkce pomocí Karnaughových map. Aplikace sestaví Karnaughovu mapu odpovídající zadané tabulce pravdivostních hodnot (získána klasickou tabulkovou metodou z hlavní formule). Tuto mapu zobrazí a po stisknutí tlačítka „OK" pak sestaví minimalizovaný tvar formule v disjunktivní nebo konjunktivní normální formě. Opět je zobrazena tabulka pravdivostních hodnot dané formule a požadovaný text je opět v okně „Výsledek". příklady jsou na obrázcích 5 a 6.

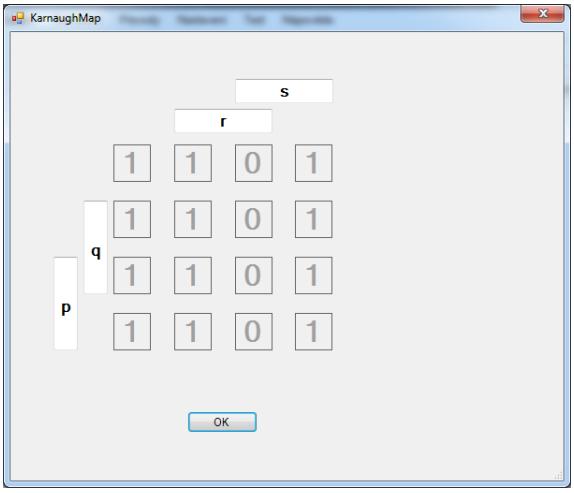
Poslední část obsahuje možnost převést danou hlavní formuli, která obsahuje pouze „klasické" výrokové spojky (viz. výše) na formule, tvořené pouze pomocí bázových spojek. Je možnost převodu na báze tvořené pouze:

1. spojkou >Sheffer< (NAND) $\Uparrow$
2. spojkou >Pierce< (Nicod, NOR) $\Downarrow$
3. spojkami negace a implikace $\neg, \Rightarrow$
4. spojkami negace a konjunkce $\neg, \wedge$
5. spojkami negace a disjunkce $\neg, \vee$

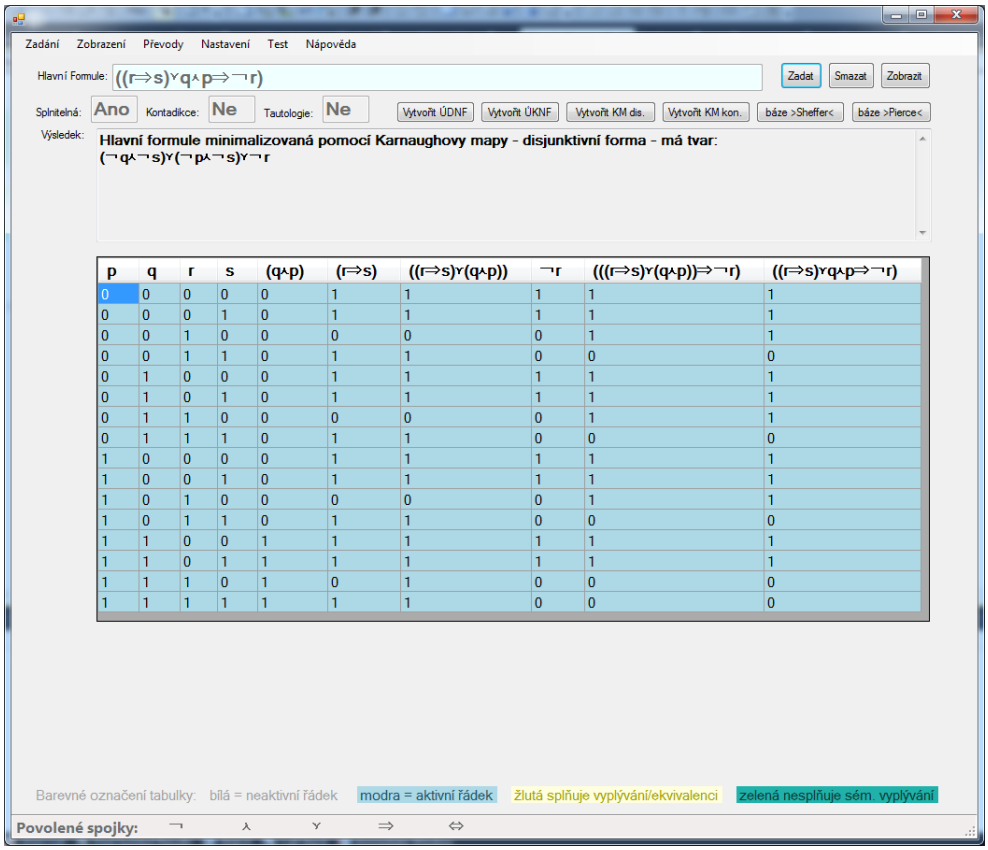
Převedená formule je také zobrazena v poli „Výsledek". Všechny tyto volby s vyjímkou posledních tří je také možné spouštět pomocí tlačítek na hlavním panelu.



Obrázek 4: Převod na bázové spojky



Obrázek 5: Karnaughova mapa



Obrázek 6: Minimalizace funkce pomocí KM

4.4.4 Nastavení

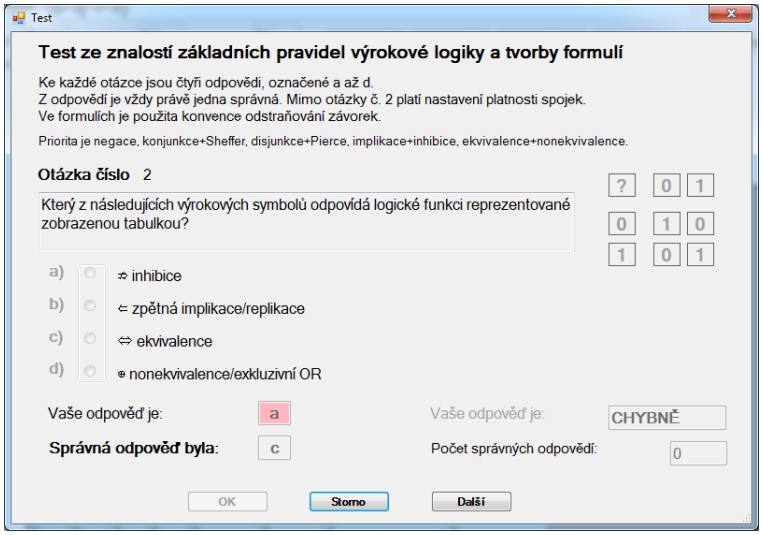
V této sekci jsou tyto možnosti:

- Označení symbolů
- Povolené spojky
- Zobrazení tabulky
- Zobrazit tlačítka sémantiky
- Zrušit tlačítka sémantiky

Tyto volby umožňují částečné přizpůsobení programu. První volba „Označení symbolů" nám umožňuje pojmenovat si podle vlastního uvážení čtyři používané výrokové symboly. Je povoleno pouze jednopísmenné označení, např. a, b, c, d. Druhá volba se nazývá „Povolené spojky". Zde je možné si vybrat, kterou z deseti výrokových spojek, odpovídající deseti booleovským funkcím dvou argumentů, bude možno používat. Platí to plně pro zadávání formulí pro zobrazení tabulek, pro tvorbu úplných normálních formulí a minimalizaci pomocí Karnaughových map. Tedy všude, kde je základem pro výpočet tabulka pravdivostních hodnot. Jiné než první čtyři „klasické" výrokové spojky nelze použít k převodu formulí na tvar obsahující pouze bázové spojky. Naopak všechny výrokové spojky mohou být použity v druhé otázce testu. Také při převodu na bázové spojky se používají i „zakázané" spojky. O povolení spojek nás trvale informuje proužek u spodního okraje obrazovky. Třetí volba „Zobrazení tabulky " dává možnost si zvolit, zda se při zobrazování tabulky pravdivostních hodnot zobrazují i řádky, které obsahuji pro výsledek nepodstatné údaje, t.j. jsou tvořeny symboly nepoužitými ve formuli představující danou tabulku. Tyto tzv. neaktivní řádky jsou bílé, ostatní řádky jsou modré. Podobně lze zamezit zobrazení sloupců reprezentující pravdivostní ohodnocení nepoužitých symbolů. Pouze u zobrazení tabulky pravdivostních hodnot jen pro jednu formuli lze pak zobrazit/skrýt sloupce s pravdivostními hodnotami jednotlivých podformulí a zobrazit/skrýt sloupec s původním tvarem formule, t.j. bez přidaných vnitřních závorek. U krátkých formulí nevyužívajících při zápisu konvenci o odstraňování závorek, budou potom poslední dva sloupce stejné.

4.4.5 Test

Touto volbou se spouští jednoduchý test o sedmi otázkách, které jsou zaměřeny na procvičení syntaxe formulí nebo znalost základních logických funkcí reprezentovaných výrokovými spojkami daného jazyka VL (právě zde není respektován „zákaz" dalších výrokových spojek, protože pak by otázka byla příliš jednoduchá). Dále je testována znalost a určování sémantické ekvivalence a sémantického vyplývání. Poslední sada otázek se týká splnitelnosti formulí a určování tautologií a kontradikcí. Příklad je možno vidět na obrázku 7.



Obrázek 7: Test znalostí

4.4.6 Nápověda

Nápověda má pouze dvě části, nazvané

- O aplikaci
- Nápověda

První volba zobrazí základní údaje o aplikaci, druhá potom rychlý přehled ovládání.

4.5 Poznámky

Poznámka na úvod: Protože aplikace vznikala ve velké časové tísni, nejsou některé z funkcí dotaženy tak, jak bych si představoval. Přesto si myslím, že tato aplikace splňuje všechny požadavky, kladené zadáním bakalářské práce.

Jak již bylo uvedeno, většinu funkcí je možné spouštět jak pomocí rozbalovací hlavní lišty, tak také pomocí tlačítek, což je daleko rychlejší. Cenou za to je jistá „nepřehlednost“ hlavní plochy. Domnívám se ale, že to po chvilce práce s aplikací přestane vadit a naopak to přinese rychlejší a snadnější ovládání. Všechny výsledky jsou prezentovány uprostřed v okně „Výsledek“. Pod ním je prostor pro tabulku pravdivostních hodnot, která je využita při většině funkcí programu. Ovládání se snaží dodržovat GUI pro Windows a tak mi mělo být pochopitelné i pro „běžné“ uživatele.

5 Programátorská dokumentace

5.1 Použitá technologie

Požadavky na systém jsou uvedeny v kapitole Systémové požadavky. Aplikace je ve formě spustitelného „.exe" souboru. Pro svoji práci potřebuje, aby na daném počítači bylo nainstalováno rozhraní .NET Framework. Odladění potom probíhalo na počítači s OS Windows 7 Professional s nainstalovaným balíčkem SP1, verze rozhraní byla .NET Framework 4.7.1. Ačkoliv aplikace nepoužívá žádné další knihovny, neměl jsem možnost ji vyzkoušet na jiné konfiguraci, takže možnost spouštění na jiné konfiguraci nemohu posoudit. Rozměry hlavního okna aplikace jsou 1050 x 900 pixelů a tomu by mělo odpovídat použité zobrazovací zařízení. Vzhledem k množství tlačítek na hlavním panelu zmenšení okna snižuje přehlednost. Na vlastní práci na aplikaci bylo použito Microsoft Visual Studio 2010. Pro tvorbu grafiky je použita technologie Windows Form, neboť nebyly použity žádné prvky, vyžadující použití WPF. Aplikace je celá napsána v jazyce C # s využitím všech výhod objektově orientovaného programování, t.j. dědičnosti, zapouzdření a polymorfismu. V podstatě všechny objekty jsou zapouzdřeny a přístup k jejich datovým složkám je pouze pomocí metod a vlastností. Výrokové spojky jsou všechny potomky jedné třídy a také prvky formulí jsou členy jedné třídy. Při tvořbě byla použita kniha [11].

5.2 Hlavní idea

Hlavní funkcí, lze říci s nadsázkou „srdcem" aplikace, je převedení textově zadané formule do podoby, která by umožnila rychloé a snadné další zpracování. Základní myšlenkou bylo, že každou formuli můžeme rozdělit tak, že jednotlivé části odpovídají buď samotnému výrokovému symbolu, negaci (opět buď symbolu, další negaci nebo závorce) nebo výrazu ohraničenému závorkou. Tyto tři základní kameny umožňují uložit jakoukoliv formuli.Pokud ohraničíme závorkou i vnější okraje formule, stává se celá formule také závorkou. K řešení daného problému byla použita rekurze, protože členy závorky nebo negace může být opět jenom negace nebo závorka nebo výrokový symbol, jehož pravdivostní ohodnocení nám musí být dodáno zvenčí. Další problém je, co je to vlastně závorka, respektive kolik může mít prvků. V okamžiku, kdybychom se drželi klasické definice formule daného jazyka VL, mohla by mít závorka jen přesně dva členy spojené výrokovou spojkou. To je sice velmi výhodné pro zjišťování pravdivostní hodnoty dané formule, ale na druhou stranu to znepřehledňuje zápis, speciálně u formulí v úplné normální formě apod. Proto je zde respektována konvence o odstraňování závorek a každá výroková spojka má svoji prioritu, tvořenou pěti úrovněmi:

1. unární
2. konjunkce+Shaffer
3. disjunkce+Pierce

4. implikace+inhibice
5. ekvivalence+nonekvivalence.

Pokud jsou „vedle sebe“ spojky se stejnou prioritou, postupuje se klasicky zleva doprava. Vyhodnocování pravdivostní hodnoty ale vždy závisí na spojení **dvou** výrokových symbolů nebo podformulí danou výrokovou spojkou a vyhodnocení logické funkce dvou argumentů, která přísluší dané spojce. To je možné řešit doplněním uzávorkování, které je ale u složitějších formulí využívajících danou prioritu trochu obtížné. Zde je použit kompromis, kdy je formule reprezentována závorkou s více členy, ale pro potřeby pravdivostního vyhodnocování a převodů na bázové spojky jsou použity tzv. „pseudozávorky“, na které je daná závorka rozdělena a které mají už vždy právě 2 členy a výrokovou spojku.

5.3 Struktura aplikace

Aplikace se skládá z následujících souborů, které tvoří daný projekt i dané řešení. Uvádím pouze mnou vytvořené soubory bez pomocných a dalších souborů, které jsou přidávány Visual Studiem.

1. Formule.cs
2. Funkce.cs
3. Hlavní_okno (Windows Form)
4. KarnaughMap.cs (Windows Form)
5. Nápověda.cs (Windows Form)
6. NastaveníTabulky.cs (Windows Form)
7. NováTabulka.cs (Windows Form)
8. OProgramu.cs (windows Form)
9. OznačníSymbolů.cs (Windows Form)
10. PovoleníSpojek.cs (Windows Form)
11. Program.cs
12. Spojky.cs
13. Symboly.cs
14. Test.cs (Windows Form)
15. Vstup.cs (Windows Form)

16. Zpracování.cs

Ačkoliv původní myšlenka byla, aby každý celek zpracovávající jeden úkol byl v samostatném souboru, ve výsledku se to zcela nepodařilo a navíc to ani v důsledku prolínání funkcionality není možné. S růstem délky kódu také trochu upadala přehlednost, ale myslím si, že strukturování aplikace zůstalo na dobré úrovni. Dále následuje popis jednotlivých souborů a tříd které obsahují.

5.3.1 Formule.cs

Obsahuje třídy, vztahující se k vnitřní reprezentaci formulí:

- class SplnitelnostFormule – obsahuje informace o splnitelnosti formulí
- class PravdivostníHodnotyPodformulí – obsahuje informace o pravdivostních hodnotách podformulí
- class TextyPodformulí – obsahuje informace o textovém vyjádření jednotlivých podformulí
- class Ohodnocení – připravuje tabulky pravdivostních ohodnocení jednotlivých výrokových symbolů
- struct Chyba – obsahuje informaci o chybě při tvorbě formulí
- struct PoložkyChyb – sdružuje jednotlivé chyby
- class PoložkyFormule – obsahuje soupis a informace o jednotlivých položkách dané formule (symbol, negace, závorka)
- class Spojky Formule – obsahuje soupis a informace o jednotlivých spojkách dané formule
- **class UniverzálníFormule** – pouze formule, ze které dědí všechny ostatní typy položek (symbol, negace,závorka). Tato třída vlastně tvoří spolu s popisem spojek „srdce“ celé aplikace.
- class FormulePseudozávorka – dědí z UniverzálníFormule, její instance se využívají při pravdivostním ohodnocení a převodech na báze. je to vlastně dílčí podformule tvořená pouze dvěma členy a spojku. Neobsahuje všechny informace, ale pouze informace o svém tvaru, pravdivostní hodnotě a převodu na báze
- class FormuleSymbol – dědí z UniverzálníFormule – instance reprezentují jednotlivé výrokové symboly
- class FormuleNegace – dědí z UniverzálníFormule – instance odpovídají jednotlivým negacím a drží a zpracovává informace o svém obsahu

- class FormuleZávorka – dědí z UniverzálníFormule – instance odpovídají celým závorkám, drží a zpracovávají jejich obsah. Při zpracování pravdivostní hodnoty a převodu na báze se dělí na PseudoZávorky.
- class CelkováFormule – přebírá některé informace z UniverzálníchFormulí, respektive jejich potomků a přidává informace o pravdivostních hodnotách a převodech na báze. Slouží i k zobrazení tabulky pravdivostních hodnot.
- class CelkovýSouhrnFormulí zachycuje informace o souboru jednotlivých instancí CelkovéFormule

5.3.2 Funkce.cs

Obsahuje třídy pro zpracování sémantického vyplývání, sémantické ekvivalence a Karnaughových map (dále označováno pouze KM).

- struct Smyčka – informace o smyčkách v KM
- class KarnaughovaMapa – obsahuje informace a metody použité při tvorbě KM až po jejich textové vyjádření – zápis formule
- classPolíčkoKarnaugh – definuje jednotlivá políčka KM, obsahuje informace potřebné při výpočtu KM. Každá instance odpovídá jednomu políčku
- class SémantickáEkvivalence – dostává potřebná data a obsahuje metody potřebné při ověřování sémantické ekvivalence až po tvorbu tabulky a textového vyjádření
- class SémantickéVyplývání – dostává potřebná data a obsahuje metody potřebné při ověřování sémantického vyplývání až po tvorbu tabulky a textového vyjádření
- class ÚplnáNormálníForma – dostává data potřebná pro vytvoření ÚKNF nebo ÚDNF a obsahuje metody pro jejich vytvoření

5.3.3 Hlavní_okno.cs

Obsahuje metody potřebné k zobrazení a obsluze událostí hlavního okna aplikace

- HlavníOkno – dědí z Form, obsahuje údaje o hlavní a dalších formulích a zajišťuje obsluhu událostí spojených s hlavním panelem. Také definuje jeho podobu. Je to zřejmě největší třída, protože metod pro obsluhu událostí je velké množství, dané velkým množstvím ovládacích prvků. Přímo v této třídě nedochází k výpočtům, ale jsou zde pomocné metody pro metody obsluhující události. Tím je docíleno toho, že zachycení a obsloužení události zvládne většinou jedna metoda, kterou je možno volat z více míst

5.3.4 KarnaughMap.cs

- class KarnaughMap – dědí z Form, velmi malá třída sloužící pouze pro zobrazení Karnaughovy mapy.

5.3.5 Nápověda.cs

- class Nápověda – dědí z Form. V podstatě jen „kontejner" pro text nápovědy. Toto je jedna z položek, které by si zasloužili dopracování.

5.3.6 NastaveníTabulky.cs

- class NastaveníTabulky – dědí z Form, velmi malá třída obsluhující nastavení parametrů pro zobrazení tabulek.

5.3.7 NováTabulka.cs

- class NováTabulka – dědí z Form, obsluhuje zadávání tabulky pomocí pravdivostních hodnot, jejich kontrolu a nastavení do systému

5.3.8 OProgramu.cs

- class OAplikaci – dědí z Form, velmi malá třída zobrazující údaje o aplikaci. V podstatě všechny hlavní součásti třídy jsou připraveny ve Visual Studiu a hlavní údaje pak v souboru AssemblyInfo.cs

5.3.9 OznačeníSymbolů.cs

- class OznačníSymbolů – dědí z Form, velmi malá třída obsluhuje nastavení textového označení výrokových symbolů. V kódu aplikace jsou důsledně odděleny jednotlivé entity od jejich textové podoby, viz.Symboly.cs

5.3.10 PovoleníSpojek.cs

- class PovoleníSpojek – dědí z Form, malá třída obsluhující povolení jednotlivých spojek. Mimo čtyř základní (implikace,ekvivalence, konjunkce a disjunkce) a samozřejmě negace je možné pro zápis formule (ne pro jejich převod na báze) použít i symboly dalších spojek, reprezentující další booleovské funkce dvou argumentů (Shefferova spojka, Piercova(Nicodova) spojka, inhibice, zpětná implikace a inhibice, nonekvivalence).

5.3.11 Program.cs

- class Program – vstupní bod programu, nejmenší třída.

5.3.12 Spojky.cs

Druhé „srdce“ programu. Podobně jako u formulí i zde je rodičem třída UniverzálníSpojka a jednotlivé spojky jsou potomky této třídy. To umožňuje práci s jednotlivými spojkami bez ohledu na jejich druh.

- **class UniverzálníSpojka** – rodičovská třída
- class Konjunkce – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky konjunkce.
- class Disjunkce – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky disjunkce.
- class Implikace – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky implikace.
- class Ekvivalence – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky ekvivalence
- class Sheffer – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky Sheffer (NAND).
- class Pierce – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky Pierce (Nicod, NOR).
- class Nonekvivalence – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky nonekvivalence (pro dvouhodnotovou logiku shodná s exkluzivní disjunkcí XOR).
- class ZpětnáImplikace – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky zpětná implikace.
- class Inhibice – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky inhibice.
- class ZpětnáInhibice – dědí z „UniverzálníSpojky“, data pro potřebu zpracování spojky zpětná inhibice.
- Další třídy – v kódu jsou definovány i další třídy, které slouží pro případné rozšíření o posledních šest booleovských funkcí se dvěma argumenty. Tyto třídy zatím nejsou využívány.
  - class SymbolSymbolA
  - class SymbolSymbolB
  - class SymbolNegaceA
  - class SymbolNegaceB
  - class SymbolVerum – konstantní jedničková funkce
  - class SymbolFalsum – konstantní nulová funkce

5.3.13 Symboly.cs

V aplikace je důsledně dbáno na **oddělení jednotlivých entit od jejich textové reprezentace.** Tato třída definuje mimo jiného i textovou podobu zobrazovaných symbolů.

- Výčtové typy - je zde definováno mnoho výčtových typů pro snadnější zapamatování a větší čitelnost kódu aplikace.
  - enum Symbol
  - enum TypSymbolu
  - enum Formule
  - enum Priorita
  - enum Boo - definuje vnitřní pravdivostní hodnoty formulí a ohodnocení jednotlivých symbolů. Na rozdíl od typu bool má tři stavy - mimo pravdy Boo.L1, nepravdy Boo.L0 definuje ještě neznámou logickou hodnotu pro nedefinovaný stav Boo.NeznámáLogickáHodnota
  - enum Přepínač
  - enum TypSmyčky
  - enum TypVýsledku
  - enum TypPřevodu
  - enum TypPožadavku
  - enum TypVýsledkuKM
- static class TiskPH - statická třída definující statickou metodu „DejString", která převádí hodnotu výčtového typu Boo na tisknutelnou podobu (0, 1, X)
- static class SpecifikaceSymbolů - statická třída definující povolení jednotlivých symbolů a spojek a definující „vlastnosti" některých výčtových typů
- static class Symboly - statická třída definující textovou podobu jednotlivých znaků a také textovou podobu pomocných znaků pro definici převodů na bázové spojky.

5.3.14 Test.cs

V tomto souboru je soustředěno vše, co souvisí s krátkým testem znalostí. Test je postaven na vzorech fomulí, do kterých jsou postupně umísťovány jednotlivé symboly a spojky, a z takto vytvářených formulí jsou pak vybírány formule splňující zkoušená kritéria, tedy splnitelnost, sémantické vyplývání a ekvivalenci apod.

- class Test - dědí z Form a obsahuje kód vztahující se k testu, t.j. zobrazení, obsluha událostí, tvorba otázek a vyhodnocení.
- class TestovacíFormule - je postavena na třídě CelkováFormule, jejíž instanci obsahuje a dále přidává údaje, které jsou dotazovány v textu.

5.3.15 Vstup.cs

Soubor obsahuje jedinou třídu, která se stará o kompletní zadání a její kontrolu po formální stránce, vlastní kontrola struktury formulí probíhá ve třídě „Zpracování".

- class Vstup – dědí z Form, obsahuje kód potřebný pro zadání aplikace a její první, tzv. „hrubou" kontrolu (mezery, počet závorek,...).

5.3.16 Zpracování.cs

Zde jsou soustředěny metody, které se starají o vlastní zpracování a vyhodnocení formule. Probíhá zde převod formule z textové podoby zpracované v instanci třídy Vstup. Je to vlastně **hlavní a nejdůležitější** část aplikace, na jejíž bezchybné práci závisí vše ostatní. Zde se vytváří vlastní formule představovaná rekurzivním stromem, kdy každá část obsahuje informaci o částech „pod sebou". Zadaná formule je tak vlastně redukována na závorku, která obsahuje informace o členech na jedné úrovni. Informace o vnořených členech v sobě obsahují vždy členové na nejbližší vyšší úrovni, což jsou vlastně jednotlivé podformule, negace a výrokové symboly. Rozklad na podformule o dvou členech pak probíhá při zpracování pravdivostní hodnoty při daném ohodnocení a pro účely převodu na bázové spojky.

- **static class Zpracování** – statická třída provádějící převod formule z textového vyjádření na stromovou strukturu

Bližší informace o jednotlivých třídách, jejich datových složkách a metodách lze získat pohledem do kódu, který jsem se snažil psát tak, aby se k němu bylo možné případně vrátit a případně ho i doplňovat a opravovat, protože časová tíseň mi nedovolila vždy vše realizovat podle mých představ. Také názvy proměnných a dalších objektů jsou voleny tak, aby bylo možno pochopit, co je jejich obsahem/funkcí.

Závěr

Tato bakalářská práce byla navržena nejen pro podporu výuky, ale zejména pro praktické procvičování základních principů výrokové logiky. Uživatel (student) si může jednoduše vyzkoušet tvorbu vlastních výrokových formulí pomocí základních , ale i běžně neužívaných výrokových spojek. Zároveň si lze zobrazit i výslednou tabulku pravdivostních hodnot při všech možných pravdivostních ohodnoceních jednotlivých výrokových symbolů. Z této tabulky je možné potom jednoduše určit, zda je daná formule splnitelná nebo zda se jedná o kontradikci nebo tautologii.

Aplikace umožňuje určit sémantické vyplývání dané formule z množiny formulí (předpokladů) nebo si zobrazit stejnou formuli pouze pomocí bázových spojek pěti různých bází. Je možné si vyzkoušet převod do úplných normálních forem i minimalizaci zadané formule pomocí Karnaughovy mapy. Výukový text je zredukován na základní minimum, protože k danému tématu, který je základním kamenem výuky informatiky a programování, existuje nepřeberné množství literatury od mnohem renomovanějších autorů. Tato aplikace se soustředí hlavně na možnost vyzkoušet si vše v praxi. Součástí je také jednoduchý test z této problematiky.

Aplikace je navržena tak, aby ji bylo možné dále doplňovat. Názvy tříd a jejich datových složek a metod jsou voleny tak, aby vyjadřovaly svoji funkci, někde i za cenu delších názvů. Je možné přidávat další spojky, které by odpovídaly všem dalším funkcím Booleovy algebry pro dva argumenty, přidávat převody formulí na formule obsahující pouze dané množiny spojek a podobně. Základy pro toto rozšíření jsou již implementovány v existujícím kódu.

Uživatelské rozhraní je navrženo tak, aby v maximální možné míře respektovalo GUI pro stolní počítače s operačním systémem Windows a zároveň byl kladen důraz na přehlednost a funkčnost jednotlivých komponent a poskytování maximálního množství možných informací.

Závěrem bych dodal, že tato práce, včetně přiložené aplikace, nemůže samozřejmě postihovat tuto širokou problematiku v úplnosti. Proto je také je napsána tak, že je možno s ní dále pracovat, případně ji i doplňovat a rozšiřovat. V žádném případě nenahrazuje hlubší studium dané problematiky, ale je primárně určena pro základní seznámení s daným tématem a pro procvičení získaných vědomostí.

Conclusions

The Bachelor's Thesis was designed not only to support teaching, but also to practise basic principles of propositional calculus. The user (student) can simply try creating custom formulas using both basic and not commonly used propositional connectives. At the same time, you can display the resulting truth table with all possible truth values of each propositional symbol. This table can be used to simply determine whether the formula is fulfillable or whether it is contradictory or tautological.

The application allows to determine the semantic result of a given formula from a set of formulas-assumptions or to view the same formula using base connectives of five different bases. It is possible to test the conversion to full normal forms and to minimize the given formula using the Karnaugh map. The lesson text is reduced to the basic minimum because there is a wealth of literature from much more renowned authors on the topic, which is the cornerstone of computer science and programming. This application focuses mainly on the ability to try everything in practice. It also includes a simple test on this issue.

The application is designed to be complemented. Class names and their data components and methods are selected to express their function, sometimes at the cost of longer names. It is possible to add additional connectives that would match all other Boolean algebra functions for two arguments, add formula conversions to formulas containing only the given set of connectives, and so on. The basics for this extension are already implemented in existing code.

The user interface is designed to respect the GUI for desktops with Windows to the greatest possible extent, while emphasizing the clarity and functionality of each component and providing the maximum amount of information possible.

Finally, I would like to say that this work, including the attached application, cannot cover this broad issue in its entirety. Therefore, it is written to make further work with it possible or to complement it and extend it. It does not, in any circumstances, replace a deeper study of the subject, but it is primarily intended for basic familiarization with the subject and for practicing the acquired knowledge.

A Obsah přiloženého CD

bin/
: Protože aplikace je ve formátu spustitelného souboru, není třeba ji instalovat a je možné ji použít i jako „Portable“. Program VYRLOGIK.EXE je spustitelný přímo z CD, takže tato sekce obsahuje jediný výše uvedený soubor.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně přílohy tvořené tímto CD. Dále jsou zde všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programu VYRLOGIK.EXE se všemi potřebnými zdrojovými texty, spolu s dalšími soubory generovanými Visual Studiem, pro bezproblémové vytvoření spustitelné verze programu.

readme.txt
: Soubor není třeba instalovat, což je napsáno i v tomto souboru.

Literatura

[1] NAVRÁTILOVÁ, Iva. *Program na podporu výuky logiky: bakalářská práce na katedře informatiky*. OLomouc: UP Olomouc, 2014. 32 s. Dostupný také z: ⟨http://www.library.upol.cz⟩.

[2] BOKR, Josef; SVATEK, Jan. *Základy logiky a argumentace: pro zájemce o umělou inteligenci, filozofii, práva a učitelství*. První vyd. Dobrá Voda u Pelhřimova: Aleš Čeněk, 2000. 173 s. ISBN 80-902627-8-3.

[3] DUŽÍ, Marie. *Logika pro informatiky: (a příbuzné obory), učební text*. První vyd. Ostrava: VŠB-Technická univerzita Ostrava, 2012, errata 2014. 183 s. Dostupný také z: ⟨http://www.cs.vsb.cz/duzi/Matlogika_ESF_Definite.pdf⟩. ISBN 978-80-248-2662-2.

[4] ANTOŠOVÁ, Marcela; DAVÍDEK, Vratislav. *Číslicová technika: učebnice*. 4, aktualizované. České Budějovice: Kopp, 2009. 305 s. ISBN 978-80-7232-394-4.

[5] *Minimalizace kombinačních funkcí pomocí Karnaughovy mapy. : učební text, automatizace 3. ročník*. Dostupný z: ⟨http://www.skola.hellebrand.cz/text0901/au.htm⟩.

[6] ŠIROKÝ, Petr. *Minimalizace logické funkce: slide*. Benešov: Integrovaná střední škola technická, 2013. Dostupný z: ⟨http://www.isstbn.cz/soubor/vvy-32-inovace-449.pdf⟩.

[7] RACLAVSKÝ, Jiří. *Úvod do logiky: klasická výroková logika*. První vyd. Brno: Masarykova univerzita, 2015. 238 s. Dostupný také z: ⟨http://is.muni.cz/publication/1298958/cs⟩. ISBN 978-80-210-7964-9 online: pdf, 978-80-210-7790-4 vázaná vazba.

[8] BĚLOHLÁVEK, Radim; VYCHODIL, Vilém. *Diskrétní matematika pro informatiky I: skripta*. Olomouc: UP Olomouc, 2006. 138 s. Dostupný také z: ⟨http://www.phoenix.inf.upol.cz⟩.

[9] BĚLOHLÁVEK, Radim. *Úvod do informatiky: skripta*. Olomouc: UP Olomouc, 2008. 64 s. Dostupný také z: ⟨http://www.phoenix.inf.upol.cz/esf/ucebni/DM1.pdf⟩.

[10] KOLAŘÍK, Miroslav. *Matematická logika: výukové slidy*. Olomouc: UP Olomouc, 2013, 2016. Dostupný z: ⟨http://www.phoenix.inf.upol.cz/~kolarikm/ML/Pred1-Pred11.pdf⟩.

[11] SHARP, John. *Visual C Sharp 2010 Krok za krokem*. První vyd. Brno: Computer Press a. s., 2010. 696 s. ISBN 978-80-251-3147-3.